(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 817 069 B1

A2

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
02.05.2003 Bulletin 2003/18

(51) Int Cl.[7]: **G06F 12/08**

(21) Application number: **97304526.3**

(22) Date of filing: **25.06.1997**

(54) **Methods and apparatus for a coherence transformer with limited memory for connecting computer system coherence domains**

Verfahren und Vorrichtung für einen Kohärenzumwandler mit begrenztem Speicher zur Verbindung von Rechnersystem-Kohärenzdomänen

Procédé et dispositif pour transformateur de cohérence avec mémoire limitée permettant la connexion des domaines de cohérence de système d'ordinateur

(84) Designated Contracting States:
DE FR GB IT NL SE

(30) Priority: **01.07.1996 US 677014**

(43) Date of publication of application:
07.01.1998 Bulletin 1998/02

(73) Proprietor: **Sun Microsystems, Inc.**
Santa Clara, California 95054 (US)

(72) Inventors:
- **Hagerstein,Erik E**
  Palo Alto CA 94043 (US)
- **Hill, Mark Donald**
  Madison WI 53705 (US)
- **Wood, David A**
  Madison WI 53705 (US)

(74) Representative: **Turner, James Arthur et al**
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(56) References cited:
US-A- 5 522 058

- **LOVETT T ET AL: "STING: A CC-NUMA COMPUTER SYSTEM FOR THE COMMERCIAL MARKETPLACE" COMPUTER ARCHITECTURE NEWS, vol. 24, no. 2, May 1996, pages 308-317, XP000592195**
- **LENOSKI D ET AL: "THE STANFORD DASH MULTIPROCESSOR" COMPUTER, vol. 25, no. 3, March 1992, pages 63-79, XP000288291**
- **O'KRAFKA B W ET AL: "AN EMPIRICAL EVALUATION OF TWO MEMORY-EFFICIENT DIRECTORY METHODS" PROCEEDINGS OF THE ANNUAL INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, SEATTLE, MAY 28 - 31, 1990, no. SYMP. 17, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 138-147, XP000144792**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 0 817 069 B1

Description

[0001] The present invention relates to a method and an apparatus for sharing memory among coherence domains of computer systems.

[0002] The sharing of memory among multiple coherence domains presents unique coherence problems. To facilitate a discussion of these coherence problems, Fig. 1 shows a computer node 100 representing, e.g., a computer node in a more complex computer system. Within computer node 100, there are shown a plurality of processing nodes 102, 104, and 106 coupled to a common bus 108. Each of processing nodes 102, 104, and 106 represents, for example, a discrete processing unit that may include, e.g., a processor and its own memory cache. The number of processing nodes provided per computer node 100 may vary depending on needs, and may include any arbitrary number although only three are shown herein for simplicity of illustration.

[0003] Within computer node 100, a common bus 108 is shown coupled to a memory module 110, which represents the memory space of computer node 100 and may be implemented using a conventional type of memory such as dynamic random access memory (DRAM). Memory module 110 is typically organized into a plurality of uniquely addressable memory blocks 112. Each memory block of memory module 110, e.g., memory block 112(a) or memory block 112(b), has a local physical address (LPA) within computer node 100, i.e., its unique address maps into the memory space of computer 100. Each memory block 112 represents a storage unit for storing data, and each may be shared among processing nodes 102, 104, and 106 via common bus 108. Of course, there may be provided as many memory blocks as desired to satisfy the storage needs of computer node 100. In some cases, many memory modules 110 may be provided by computer node 100.

[0004] As is known to those skilled in the art, computer processors, e.g., processor 116 within processing node 102, typically operates at a faster speed than the speed of the memory module 110. To expedite access to the memory blocks 112 of memory module 110, there is usually provided with each processing node, e.g., processing node 102, a memory cache 114. A memory cache, e.g., memory cache 114, takes advantage of the fact that a processor, e.g., processor 116, is more likely to reference memory addresses that it recently references than other random memory locations. Further, memory cache 114 typically employs faster memory and tends to be small, which further contributes to speedy operation.

[0005] Within memory cache 114, there exists a plurality of block frames 118 for storing copies of memory blocks, e.g., memory blocks 112. Each block frame 118 has an address portion 120 for storing the address of the memory block it cached. If the unique address of memory block 112(a) is, e.g., FF5h, this address would be stored in address portion 120 of a block frame 118 when memory block 112(a) of memory module 110 is cached into memory cache 114. There is also provided in block frame 118 a data portion 122 for storing the data value of the cached memory block. For example, if the value stored in memory block 112(a) was 12 when memory block 112(a) was cached into block frame 118, this value 12 would be stored in data portion 122 of block frame 118.

[0006] Also provided in block frame 118 is a status tag 124 for storing the state of the memory block it cached. Examples of such states are, e.g., gM, gS, and gI, representing respectively global exclusive, global shared, and global invalid. The meanings of these states are discussed in greater detail herein, e.g., with reference to Fig. 4.

[0007] A processing node may hold an exclusive copy of a memory block in its cache when it is the only entity having a valid copy. Such exclusive copy may potentially be different from its counterpart in memory module 110, e.g., it may have been modified by the processing node that cached it. Alternatively, a processing node may possess a shared, read-only copy of a memory block. When one processing node, e.g., processing node 102, caches a shared copy of a memory block, e.g., memory block 112(a), other processing nodes, e.g., processing nodes 104 and 106, may also possess shared copies of the same memory block.

[0008] If a memory block is never cached in a processing node or it was once cached but is no longer cached therein, that processing node is said to have an invalid copy of the memory block. No valid data is contained in the block frame when the state associated with that block frame is invalid.

[0009] The coherence problem that may arise when memory block 112 is shared among the processing nodes of Fig. 1 will now be discussed in detail. Assuming that processing node 102 caches a copy of memory block 112(a) into its memory cache 114 to change the value stored in memory block 112 from 12 to 13. Typically, when the value is changed by a processing node such as processing node 102, that value is not updated back into memory module 110 immediately. Rather, the updating is typically performed when memory cache 114 of processing node 102 writes back the copy of memory block 112(a) it had earlier cached.

[0010] Now suppose that before memory cache 114 has a chance to write back the changed value of memory block 112(a), i.e., 13, into memory module 110, processing node 104 wishes to reference memory block 112(a). Processing node 104 would first ascertain in its own memory cache 132 to determine whether a copy of memory block 112(a) had been cached therein earlier. Assuming that a copy of memory block 112(a) has never been cached by processing node 104, a cache miss would occur.

[0011] Upon experiencing the cache miss, processing node 104 may then proceed to obtain a copy of memory block

112(a) from memory module 110. Since the changed value of memory block 112(a) has not been written back into memory module 110 by processing node 102, the old value stored in memory block 112(a), i.e., 12, would be acquired by processing node 104. This problem is referred to herein as the coherence problem and has the potential to provide erroneous values to processing nodes and other devices that share a common memory.

[0012] Up to now, the sharing of memory blocks 112 is illustrated only with reference to devices internal to computer node 100, i.e., devices such as processing nodes 102, 104, and 106 that are designed to be coupled to common bus 108 and communicate thereto employing the same communication protocol. There may be times when it is necessary to couple computer node 100 to other external devices, e.g., to facilitate the expansion of the computer system. Oftentimes, the external devices may employ a different protocol from that employed on common bus 108 of computer node 100 and may even operate at a different speed.

[0013] External device 140 of Fig. 1 represents such an external device. For discussion purposes, external device 140 may represent, for example, an I/O device such as a gateway to a network. Alternatively, external device 140 may be, for example, a processor such as a Pentium Pro™ microprocessor (available from Intel. Corp. of Santa Clara, California), representing a processor whose protocol and operating speed may differ from those on common bus 108. As a further example, external device 140 may represent a distributed shared memory agent for coupling computer node 100 to other entities having their own memory spaces, e.g., other computer nodes having their own memory modules. Via the distributed shared memory agent, the memory blocks within computer node 100 as well as within those other memory-space-containing entities may be shared.

[0014] Although an external device may need to share the data stored in memory module 110, it is typically not possible to couple an external device, such as external device 140, directly to common bus 108 to allow external device 140 to share the memory blocks in memory module 110. The direct coupling is not possible due to, among others, the aforementioned differences in protocols and operating speeds.

[0015] An article in Computer Architecture News, Volume 24, No. 2, May 1996, pages 308-317 by Lovett T, et al and entitled "Sting: A CC-NUMA Computer System for the Commercial Market Place", describes a cache coherent non-uniform memory access (CC-NUMA multi-processor). Four processor symmetric multi-processor (SMP) nodes use a scalable coherent interface (SO)-based coherent interconnect.

The individual SMP nodes include multiple processors and memory connected via a common bus. A bridge board at each SMP node implements coherency of local and remote caches using a directory-based cache protocol. A bus-side local directory contains two bits of state information for each block in a local memory. Bus-side remote tags provide snooping information for lines in a remote cache. A network-side local memory directory is also maintained and network-side remote tags are used in the directory-based protocol.

[0016] US Patent US-A-5,522,058 describes a distributed shared memory multi-processor system capable of reducing traffic on a shared bus, without imposing constraints concerning the type of variables to be accessed in parallel programs. A plurality of processor units are coupled through a shared bus, with each processor comprising a CPU, a main memory connected with the CPU through an internal bus, a cache memory associated with the CPU, and the sharing management unit connected with the main memory and the cache memory through the internal bus. The sharing management unit includes a main memory tag memory for storing information as to whether each line of the main memory is present in the cache memory, a cache address tag memory for storing addresses estimated to be stored in an address tag memory and a cache state memory for storing the estimated cache state of cache memory. An internal access control unit controls accesses on the internal bus, an external address unit controls access on a shared bus, a main memory tag memory control unit controls read-out and updating of the main memory tag memory and a cache state memory control unit controls the read out and updating of the cache state tag memory.

[0017] In view of the foregoing, what is needed is an improved method and apparatus for permitting memory blocks having a local physical address (LPA) in a particular computer node to be shared, in an efficient and error-free manner, among interconnected entities such as other processing nodes and external devices.

[0018] An aspect of the present invention provides a method as set forth in claim 1.

[0019] Another aspect of the invention provides coherence transformer as set forth in claim 16.

[0020] Embodiments of the invention enable the efficient solving of coherence problems when memory blocks having local physical addresses (LPA) in a particular computer node of a computer system are shared by other nodes of the system as well as by external entities coupled to that computer node.

[0021] The invention will now be described by way of example with reference to the accompanying drawings, throughout which like parts are referred to by like references, and in which:

Fig. I shows, for discussion purposes, a computer node representing, e.g., a computer node in a more complex computer system.

Fig. 2 shows, in accordance with one aspect of the present invention, a coherence transformer block.

Fig. 3 shows, in accordance with one aspect of the present invention, the memory blocks and their associated memory tags (Mtags).

Fig. 4 shows, in one embodiment of the present invention, the various available states that may be stored in a Mtag.

Fig. 5 shows in greater detail, in accordance with one aspect of the present invention, the format of a typical memory access request on common bus 108.

Fig. 6 shows in greater detail, in accordance with one aspect of the present invention, the format of a typical response to the request of Fig. 5.

Fig. 7A shows, in one embodiment, the functional units within the coherence transformer.

Fig. 7B illustrates, in one embodiment, some of the external states tracked by the coherence transformer.

Fig. 8 illustrates, in accordance with one aspect of the present invention, the steps taken by the coherence transformer Fig. 8 illustrates, in accordance with one aspect of the present invention, the steps taken by the coherence transformer in response to a memory access request on the common bus..

Fig. 9 illustrates, in accordance with one aspect of the present invention, the steps taken by the coherence transformer in response to a memory access request issued by one of the external devices.

Fig. 10 illustrates, in accordance with one aspect of the present invention, the steps taken by the coherence transformer in the snoop-only mode in response to a memory access request on the common bus.

Fig. 11 illustrates, in accordance with one aspect of the present invention, the steps taken by the coherence transformer in the snoop-only mode in response to a memory access request issued by one of the external devices.

Figs. 12 and 13 illustrate, in accordance with one aspect of the present invention, the various requests and their possible responses in the Mtag-only mode.

Fig. 14 illustrates, in one embodiment of the present invention, selected transactions performed by the coherence transformer in the Mtag-only mode in response to remote memory access requests on the common bus.

Fig. 15 illustrates selected transactions performed by the coherence transformer in the Mtag-only mode in response to memory access requests from one of the external devices.

[0022] An invention is described for permitting memory blocks having a local physical address (LPA) in a particular computer node to be shared, in an efficient and error-free manner, among interconnected entities such as internal processing nodes and external devices. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be obvious, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known structures and process steps have not been described in detail in order not to unnecessarily obscure the present invention.

[0023] In accordance with one embodiment of the present invention, there is provided a coherence transformer for coupling a computer node, e.g., computer node 100, to a plurality of external devices. The coherence transformer permits an external device, which may employ a different protocol from that employed by computer node 100 and may even operate at a different speed, to access memory blocks having local physical addresses within computer node 100. In one embodiment of the present invention, the coherence transformer monitors for selected memory access requests on the bus of computer node 100. If one of the selected memory access requests on the bus of computer node 100 pertains to a memory block currently cached by an external device, the coherence transformer may provide the latest copy of that memory block to the requesting entity, thereby avoiding a coherence problem. Further, the coherence transformer also permits the external devices to coherently obtain copies of memory blocks having local physical addresses within computer node 100.

[0024] The operational details of the coherence transformer may be better understood with reference to the drawings that follow. Referring now to Fig. 2, there is provided, in accordance with one embodiment of the present invention, a coherence transformer 200 for coupling computer node 100 to one of a plurality of external devices 202, 204, and 206. Note that although only one of each type of external device (202, 204, or 206) is shown for ease of illustration, there

may in fact exist many external devices of each type coupled to coherence transformer 200. Via coherence transformer 200, the contents of the memory blocks of memory module 110, e.g., memory blocks 112, may be accessed by any of external devices 202, 204, and 206. In accordance with one aspect of the present invention, memory blocks of memory module 110 may be shared by the external devices although these external devices employ protocols and operate at speeds different from those on common bus 108 of computer node 100.

[0025] External device 202 may represent, for example, an I/O device such as a gateway to a computer network that may obtain a few memory blocks 112 at a time from memory module 110 via coherence transformer 200. External device 204 may represent, for example, a coherence domain such as a processor, whose internal protocol and operating speed may differ from that running on common bus 108. Examples of differences include differences in block sizes and signaling. External device 206 may represent, for example, a distributed shared memory agent device.

[0026] Distributed shared memory agent device 206 may include logic circuitry for connecting computer node 100 to other distributed shared memory (DSM) domains such as other computer nodes to facilitate the sharing of memory blocks among different DSM domains and with computer node 100. Further, distributed shared memory agent device 206 may permit a processing node 102 in computer node 100 to access both memory block 112 within its local memory module 110 as well as well memory blocks associated with memory modules within computer nodes 150, 160, and 170, and vice versa. The use of distributed shared memory agent 206 creates the illusion that there is a centralized shared memory resource that the processors within computer nodes 100, 150, 160, and 170 may access although this centralized memory resource is physically implemented and distributed among different computer nodes.

[0027] Coherence transformer 200 may communicate with common bus 108 of computer node 100 via a coherence transformer link 220. On the external domain, coherence transformer 200 may communicate with any of the external devices e.g., any of external devices 202, 204, and 206, via links 222, 224, and 226 using a protocol that is appropriate for the external device with which it communicates.

[0028] Referring now to Fig. 3, there are shown in memory module 110, in accordance with one aspect of the present invention, a plurality of memory tags (Mtags) 252. Each of Mtag 252 is logically associated with a memory block within memory module 110. In one embodiment, Mtags 252 are implemented in the same memory space, e.g., dynamic random access memory (DRAM), as the memory blocks with which they are associated and may be physically adjacent to its respective memory block 112. In another embodiment, Mtags 252 are logically associated with its respective memory blocks 112, albeit being implemented in a different memory space.

[0029] A Mtag 252 tracks the global state of its respective memory block, i.e., whether computer node 100 has exclusive, shared, or invalid access to a memory block (irrespective of which processing node has that memory block). Fig. 4 shows, in one embodiment of the present invention, the various available states that may be stored in a Mtag 252. In Fig. 4, three possible states are shown: gI, gS, or gM, signifying respectively that an invalid, shared, or exclusive copy of a memory block is being held by internal entities, i.e., entities within computer node 100. Note that for the purposes of the present invention, the state of a Mtag 252 is determined by whether its associated memory block is referenced by internal entities (e.g., by memory module 110 or any of processors 102, 104, and 106) or by devices in the external domain (i.e., external to computer node 100 such as any of external devices 202, 204, and 206). Further, the state of each Mtag is generally independent of which specific device within these domains currently has the memory block. Consequently, a Mtag can generally indicate whether an external device has a valid copy of a memory block. The state of Mtag generally cannot indicate which device, either internally or externally, currently has the latest valid copy.

[0030] If the state of Mtag 252 is gM, the internal domain has a valid, exclusive (and potentially modified from the copy in memory module 110) copy of the associated memory block. Further, there can be no valid (whether exclusive or shared) copy of the same memory block in the external domain since there can be no other valid copy of the same memory block existing anywhere when an exclusive copy is cached by a given device. If the state of Mtag 252 is gS, the internal domain has a valid, shared copy of the associated memory block. Further, since many shared copies of the same memory block can exist concurrently in a computer system, the external domain may have other shared copies of the same memory block as well. If the state of Mtag 252 is gI, the internal domain does not have a valid copy of the associated memory block. Since neither memory module 110 nor any bus entities 102, 104, and 106 has a valid copy, the valid copy may reside in the external domain. In one embodiment, when the state of Mtag 252 is gI, it is understood that the external domain has an exclusive (and potentially modified) copy of the associated memory block.

[0031] Fig. 5 shows in greater detail in accordance with one aspect of the present invention the format a memory access request 400, representing a typical memory access request on common bus 108. The memory access request may be output by, for example, one of the processing nodes 102, 104, or 106 or by coherence transformer 200 on behalf of one of the external devices 202, 204, or 206.

[0032] Memory access request 400 typically includes a type field 402, an address field 404, a source ID field (SID) 406, and an own flag 408. Type field 402 specifies the type of memory access request being issued. As will be discussed in detail in connection with Fig. 8 herein, memory access request types specified in field 402 may include, among others, a request to own (RTO), remote request to own (RRTO), request to share (RTS), remote request to share

(RRTS), and write back (WB). Address field 404 specifies the address of the memory block being requested by the progenitor of memory access request 400. Source ID field 406 specifies the identity of the progenitor of memory access request 400, i.e., the entity that issues memory access request 400.

[0033] Own flag 408 represent the flag bit that is normally reset until one of the entities other than memory 110 that is capable of servicing the outstanding memory access request, e.g., one of processing nodes 100-106, sets own flag 408. An entity coupled to common bus 108 may wish to set own flag 408 to indicate that the current memory access request should not be serviced by memory module 110, i.e., one of the entities capable of caching that memory block had done so and may now potentially have a newer copy than the copy in memory module 110.

[0034] Fig. 6 shows in greater detail in accordance with one embodiment of the present invention the format of a response 500. Response 500 is typically issued by the entity responding to an earlier issued memory access request, e.g., one having the format of memory access request 400 of Fig. 5. As is shown in Fig. 6, response 500 includes a source ID (SID) field 502, representing the unique ID of the requesting entity to which the response should be sent. In one embodiment, the content of SID field 502 is substantially similar to the SID data contained in source ID field 406 of Fig. 4. The use of the source ID permits coherence transformer 200 to communicate directly with common bus 108 and entitles coherence transformer 200 to rely on the mechanism of common bus 108 to forward the response, using the SID, to the appropriate final destination. Response 500 further includes a data field 504, representing the content of the relevant memory block.

[0035] Fig. 7A shows, in one embodiment, the functional units within coherence transformer 200. In one embodiment, the functional units are implemented as digital logic circuits. As can be appreciated by those skilled in the art, however, these functional units may be implemented either in hardware (digital or analog) or in software, depending on needs. Within coherence transformer 200, there is shown in coherence transformer 200 a tag array 250, representing the mechanism for keeping track of the memory blocks accessed by a device on the external side, e.g., one of external devices 202, 204, and 206. Within tag array 250, there is shown a plurality of tags 273, 274, 276, 278. In one embodiment, there may be provided as many tags in tag array 250 as reasonably possible. As will be discussed in greater detail later, the provision of a large number of tags in snoop tag array 250 advantageously minimizes any impact on the bandwidth of common bus 108 when a large number of memory blocks are cached by the external domain. Of course, the number of tags in tag array 250 may vary depending on needs and may represent any arbitrary number.

[0036] In accordance with one embodiment of the invention, externally cached memory blocks are tracked, for the duration that they are externally cached, in tags within snoop tag array 250 whenever possible. When tags run out, i.e., when there are more memory blocks currently cached externally than there are available tags within snoop tag array 250, the embodiment advantageously permits the extra memory blocks to be externally cached without tracking them in snoop tag array 250 for the entire duration that they are externally cached.

[0037] As will be described in detail herein, buffer 280 represents a tag especially dedicated for temporarily storing memory blocks that are going to be cached externally without being tracked in snoop tag array 250 (referred herein as the Mtag-only approach). In other words, the purpose of buffer 280 is to temporarily track memory blocks cached externally in accordance with the Mtag-only approach and while in transit. Once that memory block is properly cached externally using the Mtag-only approach, e.g., by writing back to memory module 110 the proper Mtag state, buffer 280 may be recycled to temporarily track another memory block externally cached using the Mtag-only approach and while in transit. In one embodiment, multiple buffers 280 may be provided to track memory blocks in transit and cached externally using the Mtag-only approach.

[0038] Note that buffer 280 does not track an externally cached memory block for the entire duration that that memory block is cached externally. In other words, buffer 280 may be recycled for reuse in temporarily storing the identity of another externally tracked memory block that is in transit and externally cached using the Mtag-only approach even if the last memory block it temporarily stored is still being cached externally. This is different from the function provided by tags of snoop tag array 250, e.g., tags 273, 274, 276, and 278, which track externally cached memory blocks for the entire duration that they are cached externally, and are only recycled for reuse when the memory blocks they track are no longer externally cached. Because of its temporary storage purpose, buffer 280 may not be counted, in one embodiment, as part of the tags available for tracking externally cached blocks since buffer 280 may be used for temporary storage only. The operation of buffer 280 will be described in detail herein.

[0039] There is coupled to tag array 250 a snooping logic 260, representing the circuitry employed to monitor memory access requests on common bus 108 on Fig. 1. In one embodiment, snooping logic 260 is substantially similar to the conventional snooping logic employed in each of processing nodes 102, 104, and 106 for permitting those processing nodes to monitor memory access requests on common bus 108.

[0040] Within each tag, e.g., tag 273, there is a state field 272(a), an address field 272(b), and an optional valid flag 272(c). Optional valid flag 272(c) indicates whether a tag is allocated or is empty. In one embodiment, state field 272 (a) may store one of the three states (eM, eS, and eI) although additional states may be employed if desired. Fig. 7B shows, in one embodiment of the present invention, the various available states that may be stored in state field 272 (a) of the tags of tag array 250. In Fig. 7B, three possible external states are shown: eI, eS, and eM, signifying respec-

tively that an invalid, shared, and exclusive copy of a memory block is being cached by an external device. Note that these external states are different from the global states in that the external states reflect the states of the memory block from the perspective of the external domain. On the other hand, the global states (reflected in the Mtags) reflect the states of the memory blocks from the perspective of the internal domain. Address field 272(b) stores the address of the memory block cached, thereby permitting coherence transformer 200 to track the memory blocks that both are currently cached by an external device and tracked within snoop tag array 250.

[0041] It should be apparent to those skilled in the art from the foregoing that some type of protocol conversion may be necessary to permit devices and systems utilizing different protocols and/or operating at different speeds to share memory blocks. Protocol transformer logic 262 represents the circuitry that permits coherence transformer 200 to communicate with an external device, e.g., one of external devices 202, 204, and 206. Protocol transformer logic 262 may be omitted, for example, if the external device employs the same protocol as that employed in computer system 100 or operates at the same speed. Keep in mind that the specific protocol employed to communicate with a specific external device may vary greatly depending on the specification of the protocol employed by that external device. As will be discussed in greater detail herein, it is assumed that communication for the purpose of sharing memory blocks with the external devices can be accomplished using a generalized protocol known as the X-protocol. The adaptation of the described X-protocol to a specific external device should be readily apparent to those skilled in the art given this disclosure.

[0042] In one embodiment of the present invention, a coherence transformer tracks as many of the memory blocks cached by the external device as it can in the tags of snoop tag array 250. As will be discussed in detail later herein, whenever coherence transformer 200 can track an externally cached memory block in a tag within snoop tag array 250 for the entire duration of that the block is externally cached, i.e., there is room in snoop tag array 250 for such tracking, there is no need to change the Mtag state of that memory block within memory module 110. Advantageously, when the system operates in the snoop-only approach, adverse impact on common bus 108 is minimized since there is no need to take up the bandwidth of common bus 108 to perform a write to memory module 110 to change the Mtag of externally cached memory blocks.

[0043] On the other hand, when there is no more room in snoop tag array 250 for such tracking, the embodiment still allows a memory block within memory module 110 to be externally cached. Instead of tracking this externally cached memory block in snoop tag array 250 for the entire duration of that the block is externally cached, however, coherence transformer 200 merely temporarily track this memory block in a buffer that is especially reserved for this purpose, e.g., buffer 280, and writes the new Mtag back into memory 110 at the earliest opportunity. Once the new Mtag is written into memory 110, there is advantageously no need to continue to track this memory block, and buffer 280 can be made available again (via a flag, for example) to temporarily track another externally cached memory block.

[0044] The coherence transformer then monitors memory access requests on the bus of computer system 100. If one of the memory access requests on the bus of computer system 100 pertains to a memory block currently cached by an external device and tracked in snoop tag array 250(whether in tags 273-278 or temporarily in buffer 280), the coherence transformer enters the snoop-only mode. As the term is used herein, the snoop-only mode pertains to the mode wherein coherence transformer 200, having found a match between the requested memory block and one of the memory blocks tracked in snoop tag array 250, intervenes to provide the latest copy of that memory block, instead of allowing memory module 110 to respond to the outstanding memory access request. In this manner, coherence problems are advantageously avoided.

[0045] If the outstanding memory access does not pertain to a memory block currently tracked in a tag of snoop tag array 250 (whether in tags 273-278 or temporarily in buffer 280), the requested memory block is either not cached by an external device, or is currently cached by an external device but the Mtag state in memory module 110 has been modified, coherence transformer 200 does nothing for the moment. This is because there would be no harm in allowing in allowing memory module 110 to respond and to subsequently handle this memory access request using the Mtag only approach.

[0046] Fig. 8 illustrates, in accordance with one embodiment of the present invention, the steps involved for coherence transformer 200 to respond to a memory access request outstanding on common bus 108, i.e., one originated in the internal domain. In step 502, a memory access request appears on common bus 108 and seen by coherence transformer 200, which is coupled thereto. In step 504, coherence transformer ascertains whether the requested memory block, i.e., the one requested in the internally originated memory access request on common bus 108, matches one of the memory blocks tracked by the tags in snoop tag array 250.

[0047] If there is a match, the method proceeds to step 506 wherein outstanding memory access request on common bus 108 is handled in accordance with the snoop-only approach (described in detail in section A herein). On the other hand, when there is not a match, the method proceeds to step 508 wherein outstanding memory access request on common bus 108 is handled in accordance with the Mtag-only approach (described in detail in section B herein).

[0048] Fig. 9 illustrates, in accordance with one embodiment of the present invention, the steps involved for coherence transformer 200 to handle a memory access request originated in the external domain, i.e., issued by one of the

external devices. In step 604, coherence transformer 200 receives from the external device a command to cache a specific memory block within memory module 110. In step 606, coherence transformer ascertains whether there is room in its snoop tag array 250 to track this memory block for the entire duration of its being externally cached. In one embodiment, the check in step 606 involves determining whether there is one unused tag, beside the buffer(s) set aside for temporary storage of memory blocks in transition when all other tags are full, to track the externally requested memory block.

[0049] If there is room in snoop tag array 250 to track this memory block for the entire duration of its being externally cached, the method proceeds to step 608 wherein the external memory access request is handled in accordance to the snoop-only approach (described in detail in section A herein).

[0050] On the other hand, if there is no room to track this externally requested memory block for the entire duration of its being externally cached, the method proceeds to step 610 wherein the external memory access request is handled in accordance to the Mtag-only approach (described in detail in section B herein).

[0051] To increase the likelihood that a new externally originated request can advantageously employ the snoop-only approach (thereby saving the bandwidth of common bus 108), the coherence transformer may opt to select to replace a snoop tag (e.g., 273 in Fig. 7A). In other words, the coherence transformer may opt to unallocate a snoop tag although the memory block it tracks is still cached by an external device. To do this, the coherence transformer writes the block's Mtag into memory module 110 as a function of the current external state. For example, if the external state is eM, the Mtag should be set to gI, eS sets Mtag to gS, and eI sets Mtag to gM. The algorithm employed to select which tag to replace may follow any conventional cache replacement algorithm, e.g., least recently used (LRU), random, first-in-first-out (FIFO), or the like.

[0052] The unallocating of tags may, in some cases, be particularly advantageous, especially if the newly requested block is more active than the old one. Thus, the bandwidth on common bus 108 saved on the new requests may exceed the bandwidth consumed in writing the old block's Mtag back to memory module 110.

Section A: Snoop Only approach

[0053] In the snoop only approach, each externally cached memory block is tracked in a tag in snoop tag array 250. The externally cached memory block, once tracked, will continue to be tracked until the cached memory block is written back into memory module 110, thereby freeing up the tag to track another externally cached memory block.

[0054] When there is a memory access request, e.g., one having a format of memory access request 400 of Fig. 4, on common bus 108, coherence transformer 200 (via coherence transformer link 220) monitors this memory access request and checks address field 404 of the memory access request against the addresses of the memory blocks cached by one of the external devices. With reference to Fig. 10A, this checking is performed, in one embodiment, by comparing address field 404 against the addresses stored in address fields 272(b) of the tags within the tag array 250.

[0055] If there is an address match, the state of the matched tag is then checked to ascertain whether the memory block cached by the external device is of the appropriate type to service the outstanding memory access request. This is because an external device may currently have only an invalid copy of a memory block and would therefore be incapable of servicing either a RTO or a RTS memory access request.

[0056] If the state of the matched tag indicates that the externally cached memory block is the appropriate copy for servicing the outstanding memory access request, snooping logic 260 of coherence transformer 200 may then set own flag 408 to signify that the default response should be overridden, i.e., memory module 110 should not respond to the outstanding memory access request since there may be a more current copy cached by one of the external devices.

[0057] Own flag 408 of memory access request 400, while logically associated with the memory access request, is skewed in time therefrom in one embodiment. In this manner, the own flag may arrive a few cycles later than the rest of the memory access request to allow time for entities, such as coherence transformer 200, to ascertain whether they should respond to the memory access request with a more recent copy of the requested memory block than that available in memory module 110.

[0058] Coherence transformer 200 then obtains the appropriate copy of the requested memory block from the external device, using its knowledge of which external device currently holds the most recent copy. Coherence transformer 200 then formulates a response 500 to return the appropriate copy of the requested memory block to common bus 108 to be forwarded to the requesting entity, i.e., one identified by the source ID in the issued memory access request.

[0059] As mentioned earlier, whenever memory block 112 is cached by one of the external devices, a tag is employed in tag array 250 of coherence transformer 200 to track the fact that this memory block is being externally cached, and also the external state of that memory block. In this manner, coherence transformer 200 can keep track of which memory block of computer system 100 has been cached by the external devices and (in state field 272(a) of the matching tag) which type of copy was actually cached.

[0060] When the state in the tag is eI, either the external devices do not have a copy of the requested memory block (even if there is a match between the incoming address and one of the addresses stored in tag array 250) or one of

the external devices does have a copy but this memory block is not tracked by snoop tag array 250 since its corresponding Mtag state has already been properly reflected in memory module 110. In this case, the invention advantageously treats the requested memory block as if it is not cached by the external domain, and simply ignore the present memory access request.

[0061] If the state in the matching tag is eS, at least one of the external devices has owned a shared, read-only copy of the requested memory block. If the state in the matching tag is eM, one of the external devices owns an exclusive copy of the requested memory block, which it can use to respond to, for example, a RTO memory access request. Further, the external device owning the exclusive copy can unilaterally modify this copy without having to inform other bus entities attached to common bus 108.

[0062] The operation of coherence transformer 200 may be more clearly understood with reference to Figs. 10 and 11. Fig. 10 illustrates, in one embodiment of the present invention, selected transactions performed by coherence transformer 200 in response to memory access requests on common bus 108.

A1. RTO Request on Bus

[0063] Referring now to Fig. 10, when a RTO memory access request is issued by one of the bus entities on common bus 108 (as the term is used hereinafter, a "bus entity" refers to any entity such as a processing unit or any other device that is coupled to common bus 108 for sharing a memory block), this RTO memory access request is forwarded to all bus entities, including coherence transformer 200. Coherence transformer 200 then ascertains whether the address of the requested memory block matches one of the addresses stored in tag array 250 of coherence transformer 200.

[0064] If there is an address match, the current state of the matching tag is then ascertained to determine whether the copy cached by one of the external devices is of the appropriate type for responding to the memory access request on common bus 108. If the memory access request is a request for an exclusive copy of a memory block (a RTO) or a request for a shared copy of a memory block (a RTS), and the current state of the matching tag is eI (invalid), coherence transformer 200 ignores the present RTO memory access request since the external device either never cached the requested memory block or one of the external devices does have a copy but this memory block is not tracked by snoop tag array 250 since its corresponding Mtag state has already been properly reflected in memory module 110.

[0065] If the memory access request on common bus 108 is a RTO (the first RTO of this transaction) and the current tag is eS, coherence transformer 200 needs to invalidate any shared external copy or copies currently cached by one or more of the external devices. This invalidation is illustrated in Fig. 10 by the XINV command, which is a X-protocol invalid command directed at every external device currently having a shared external copy. Following the invalidation, the new state of the memory block in the external device is invalid (New State = eI).

[0066] Upon confirmation that the external device has invalidated its shared copy of the requested memory block (via the X-protocol command XINV_ack), coherence transformer 200 then downgrades the state of the matching tag to invalid (New State = eI) to reflect the fact that there is no longer a valid external copy. Coherence transformer 200 then obtains a copy of this requested memory block from computer system 100 and invalidates all internal copies cached by bus entities within computer system 100. Both these actions are accomplished when coherence transformer 200 issues a RTO command (the second RTO of this transaction) to common bus 108 and receives the requested data (via the RTO_data response to the second RTO). The copy of the requested memory block is then sent to common bus 108 to be forwarded to the entity that originally issues the RTO memory access request (via the RTO_data response to the first RTO). In one embodiment, the coherence transformer passes Mtag gM with the final RTO data. Alternatively, the coherence transformer may first update the Mtag in memory module 110 and then respond to the RTO with data.

[0067] Note that the use of the XINV command advantageously invalidates all shared copies of the requested memory block cached by the external device(s). Further, the use of the RTO request by coherence transformer 200 to common bus 108 advantageously ensures that all shared copies within computer system 100 are invalidated and obtains the required memory block copy to forward to the requesting entity.

[0068] The current state of the matching tag may be an eM when a RTO memory access request appears on common bus 108. The eM state signifies that an external device currently caches an exclusive (and potentially modified) copy of the memory block being requested. In this case, coherence transformer 200 may obtain the exclusive (and potentially modified) copy of the requested memory block from the external device and return that copy to the entity that originally issues the RTO request on common bus 108. In one embodiment, the coherence transformer passes Mtag gM with the final RTO data. Alternatively, the coherence transformer may first update the Mtag in memory module 110 and then respond to the RTO with data.

[0069] As shown in Fig. 10, coherence transformer 200 may issue a RTO-like transaction using the X-protocol XRTO transaction to request the exclusive copy of the memory block, which is currently being cached by one of the external devices. If there are multiple external devices coupled to coherence transformer 200, there may be provided with coherence transformer 200 conventional logic, in one embodiment, to allow coherence transformer 200 to determine

which external device currently holds the desired exclusive copy of the requested memory block.

[0070] The requested copy of the memory block is then returned to coherence transformer 200 from the external device that currently holds it (using the XRTO_data command, which is analogous to the aforementioned RTO_data except accomplished using the X-protocol). Further, the external copy that was previous cached by the external device is downgraded to an invalid copy. This downgrade is tracked in the matching tag in tag array 250, thereby changing the state to el (New State = el). After coherence transformer 200 receives the exclusive copy of the requested memory block from the external device that previously cached it, coherence transformer 200 formulates a response to the original RTO, using e.g., using an RTO_data response in a format similar to that shown in Fig. 5, to furnish the requested exclusive copy of the memory block to common bus 108 to be forwarded to the entity that originally issued the RTO memory access request. In one embodiment, the coherence transformer passes Mtag gM with the final RTO data. Alternatively, the coherence transformer may first update the Mtag in memory module 110 and then respond to the RTO with data.

A2. RTS Request on Bus

[0071] If the memory access request on common bus 108 represents a request for a shared, read-only copy of a memory block, i.e., a RTS (the first RTS) and the current state of the matching tag is el (invalid), coherence transformer 200 will ignore the outstanding RTS memory access request even if there is a match between the incoming address and one of the addresses stored in the tags of tag array 250.

[0072] On the other hand, if the current state of the matching tag is eS (i.e., one or more of the external devices currently cache shared, read-only copies of the requested memory block), coherence transformer 200 may, in one embodiment, obtain the shared, read-only copy of the requested memory block from computer system 100 itself, e.g., by issuing a RTS request to common bus 108 (the second RTS request). After coherence transformer 100 receives the shared, read-only copy from computer system 100 (via the RTS_data response to the second RTS), it then forwards the copy to common bus 108 to be forwarded to the bus entity that originally issues the RTS. command (via the RTS_data response to the first RTS).

[0073] If the memory access request on common bus 108 is a RTS and the current state of the matching tag is eM, coherence transformer 200 may obtain the copy of the memory block that is currently exclusively owned by one of the external devices. Further, coherence transformer 200 may downgrade that external copy to a shared copy, and return the data to common bus 108 to be forwarded to the entity that originally issued the RTS memory access request. To accomplish the foregoing, coherence transformer 200 may issue a X-protocol RTS-like transaction (XRTS) to the external device that currently exclusively owns the requested memory block. That external device will return the copy it previously owns as an exclusive copy to coherence transformer 200 (XRTS_data) and also downgrade the external copy from an exclusive copy to a shared .copy (New State = eS in the matching tag). When coherence transformer 200 receives the copy of the memory block from the external device, it can forward that copy to common bus 108 (via the RTS_data command) to be forwarded to the entity that originally issue the RTS memory access request. The coherence transformer may then write the data and Mtag gS into memory module 110 with the WB transaction.

A3. WB Request on Bus

[0074] If the memory access request on common bus 108 represents a write back (WB) request, i.e., signifying that a bus entity coupled to common bus 108, other than coherence transformer 200, wishes to write back the exclusive copy of the memory block it currently owns. In this situation, the response of coherence transformer 200 depends on the state of the copy of the memory block currently cached by the external device. Generally, the entity that issues the write back memory access request owns the exclusive copy of that memory block, and any copy that may have been cached by an external device earlier must be invalid by the time the write back memory access request is asserted by its owner on common bus 108. Consequently, the current state in the matching tag, if any, should be el (invalid), in which case coherence transformer 200 does nothing and ignores the outstanding write back transaction on common bus 108.

[0075] If, for some reason, the current state of a matching tag is eS or eM, an error condition would be declared since there cannot be another shared or exclusive copy in the network if the write back entity already has an exclusive copy of the memory block. The resolution of this error condition is conventional and may include, for example, flagging the error and performing a software and/or a hardware reset of the system.

[0076] Coherence transformer 200 not only interacts with the processing nodes within computer system 100 to respond to memory access requests issued by those processing nodes, it also interacts with the external devices, e.g., one of external devices 202, 204, and 206, in order to service memory access requests pertaining to memory blocks having local physical addresses within computer system 100. Fig. 11 illustrates, in accordance with one embodiment, selected transactions performed by coherence transformer 200 in response to memory access requests from one of

the external devices.

[0077] In Fig. 11, the memory access requests are issued by one of the external devices, e.g., one of devices 202, 204, or 206, to coherence transformer 200. If another external device currently caches the required copy of the requested memory block, this memory access request may be handled by logic circuitry provided with coherence transformer 200 without requiring the attention of common bus 108.

[0078] On the other hand, if another external device does not have the valid copy of the requested memory block to service the external memory access request, coherence transformer 200 then causes a memory access request to appear on common bus 108, using a protocol appropriate to computer system 100, so that coherence transformer 200 can obtain the required copy of the requested memory block on behalf of the requesting external device. With reference to Fig. 9, the snoop-only approach to XRTO, XRTS, and XWB of this section A assumes that there is room in snoop tag array to track the memory block to be externally cached for the entire duration that this memory block is externally cached (step 608 of Fig. 9). If there is not enough room, the embodiment preferably employs the Mtag-only approach to handle the memory requests externally originated (step 610 of Fig. 9).

[0079] In the remaining discussion of section A, since a copy of the memory block is now cached by an external device and serviced in accordance with the snoop-only approach, this memory block is tracked in a tag in tag array 250 of coherence transformer 200.

[0080] In one embodiment, the coherence transformer always asserts the own flag on bus transactions for blocks that are externally cached as exclusive or may be externally cached as shared. This advantageously allows the coherence transformer to take more time to correctly handle such requests.

A4. XRTO Request

[0081] Referring now to Fig. 11, when an external device issues a memory access request to obtain an exclusive copy of a memory block having a local physical address within computer system 100, e.g., memory block 112(a), coherence transformer 200 first determines whether the address of the requested memory block matches one of the addresses stored in tag array 250 of coherence transformer 200. If there is a match, the current state of the tag that matches the incoming address, i.e., the matching tag, is then ascertained to determine whether an external device, e.g., any of the external devices that couple to coherence transformer 200, has cached a copy of the requested memory block.

[0082] If the current state of the matching tag is eI (invalid), coherence transformer 200 proceeds to obtain the requested memory block from common bus 108. This is because a current state invalid (eI) indicates that either none of the external devices currently caches a valid (whether a shared, read-only copy or an exclusive copy) of the requested memory block or that an external device is currently caching a valid copy of the requested memory block but this fact is not tracked in snoop tag array 250 since the Mtag corresponding to the requested memory block has already been properly updated in memory module 110. In this case, the embodiment advantageously treats the memory block as if it is not cached by one of the external devices, thereby permitting coherence transformer to request this memory block from the internal domain.

[0083] Further, since the requested memory block will be cached by the requesting external device, e.g., I/O device 202, after the current memory access request is serviced (since it has already been ascertained in step 606 of Fig. 9 that there is room in snoop tag array 250), this requested memory block needs to be tracked within tag array 250 of coherence transformer 200 so that the next memory access request pertaining to this memory block can be serviced by coherence transformer 200 on behalf of the external device, e.g., I/O device 202, which then has an exclusive (and potentially modified) copy. An unused tag in tag array 250, e.g., one of the tags has a current state invalid or is simply unused, may be employed for tracking the newly cached memory block along with the state of the copy (e.g., eM, eS, or eI). Fig. 11 shows this situation wherein the old tag has state eI.

[0084] Referring back to the case in Fig. 11 where there exists a XRTO memory access request from an external device and the current state of the matching tag is eI or there is no tag that matches, coherence transformer 200 acts as another bus entity coupled to common bus 108, i.e., it communicates with common bus 108 using a protocol appropriate to computer system 100 to issue a memory access request for an exclusive copy of the requested memory block. In other words, coherence transformer simply issues a RTO memory access request to common bus 108.

[0085] The presence of the request-to-own (RTO) memory access request on common bus 108 causes one of the bus entities, e.g., one of processing nodes 102, 104, and 106, or memory module 110, to respond with the latest copy of the requested memory block (RTO_data transaction in Fig. 11). After coherence transformer 200 receives the exclusive copy of the requested memory block from common bus 108, it then forwards this exclusive copy to the requesting external device using a protocol that is appropriate for communicating with the requesting external device (generalized as the X-protocol XRTO_data command herein). Further, the new state of the tag that tracks this requested memory block is now upgraded to an eM state, signifying that an external device is currently caching an exclusive (and potentially modified) copy of this memory block.

[0086] If one of the external devices, e.g., I/O device 202, issues a read-to-own memory access request (using the X-protocol XRTO) for a given memory block and a shared, read-only copy of that memory block has already been cached by a sister external device, e.g., coherent domain device 204, there would already be a tag in tag array 250 for tracking this memory block. However, the state of such tag will reflect an eS copy since the sister external device only has a shared read-only copy. In this case, there is no need to allocate a new tag to track the requested memory block. Coherence transformer must still invalidate all other shared copies of this memory block in computer system 100 and on the sister external devices, as well as upgrade the state of the matching tag to an eM state.

[0087] To invalidate the shared copies at the sister external devices, coherence transformer 200 may issue an invalidate command (XINV) to those sister external devices and wait for the acknowledged message (XINV_ack). To invalidate shared, read-only copies of the requested memory block on the bus entities in computer system 100, coherence transformer 200 issues a request-to-own (RTO) memory access request to common bus 108. This RTO command both obtains a copy of the requested memory block (RTO_data transaction) and invalidates the shared, read-only copies cached by the bus entities in computer system 100.

[0088] After coherence transformer 200 receives the copy of the requested memory block from common bus 108 (via the RTO_data transaction), coherence transformer 200 may then forward this copy to the requesting external device to service the XRTO memory access request (XRTO_data transaction). Further, the state associated with the matching tag in tag array 250 may be upgraded from an eS (shared) state to an eM (exclusive) state.

[0089] If the memory access request received by coherence transformer 200 is a request for an exclusive memory block (XRTO) from an external device and a sister external device is currently caching the exclusive copy of that memory block, logic circuitry provided with coherence transformer 200 preferably obtains the requested memory block from the sister external device to satisfy the XRTO request without requiring the attention of coherence transformer 200 itself. As a general rule, if there are more than one external devices, they may, in one embodiment, resolve memory access requests by passing copies of memory blocks among themselves before asking for it from common bus 108 (via coherence transformer 200). On the other hand, if the XRTO memory access request for a memory block comes from an external device that already is currently caching the exclusive copy of the same requested memory block, an error condition exists as shown in Fig. 11. The error condition may be handle using a variety of conventional techniques, e.g., flag the error and/or perform a software or hardware reset. Further, in one embodiment, the coherence transformer could handle XRTO's to externally cache blocks by forwarding requests to sibling devices.

A5. XRTS Request

[0090] When an external device issues a memory access request to obtain a shared, read-only copy of a memory block having a local physical address within computer system 100 such as memory block 112(a) (via an XRTS command), coherence transformer 200 first determines whether the address of the requested memory block matches one of the addresses stored in tag array 250 of coherence transformer 200. If there is a match, the matching tag is then ascertained to determine whether an external device, e.g., any of the external devices that couple to coherence transformer 200, has cached a copy of the requested memory block.

[0091] If the current state of the matching tag is el (invalid), coherence transformer 200 proceeds to obtain the requested memory block from common bus 108. This is because a current state invalid (el) indicates that none of the external devices currently caches a valid (whether a shared, read-only copy or an exclusive copy) of the requested memory block or that an external device is currently caching a valid copy of the requested memory block but this fact is not tracked in snoop tag array 250 since the Mtag corresponding to the requested memory block has already been properly updated in memory module 110. In this case, the embodiment advantageously treats the memory block as if it is not cached by one of the external devices, thereby permitting coherence transformer to request this memory block from the internal domain.

[0092] Further, since the requested memory block will be cached by the requesting external device, e.g., I/O device 202, after the current memory access request is serviced, this requested memory block needs to be tracked within tag array 250 of coherence transformer 200 (step 606 already ascertained that there is room for tracking the requested memory block in snoop tag array 250). An unused tag in tag array 250, e.g., one of the tag has a current state invalid or is simply unused, may be employed for tracking the newly cached memory block, along with the state of the copy (i.e., eS).

[0093] Referring back to the case in Fig. 11 where there exists a XRTS memory access request from an external device and the current state of the matching tag is el or there is no tag that matches, coherence transformer 200 may act simply as another bus entity coupled to common bus 108, i.e., it communicates with common bus 108 using a protocol appropriate to computer system 100 to issue a memory access request for a shared, read-only copy of the requested memory block. In other words, coherence transformer simply issues a RTS memory access request to common bus 108.

[0094] The presence of the request-to-share (RTS) memory access request on common bus 108 causes one of the

bus entities, e.g., one of processing nodes 102, 104, and 106, or memory module 110, to respond with the shared copy of the requested memory block (RTS_data transaction in Fig. 11). After coherence transformer 200 receives the shared, read-only copy of the requested memory block from common bus 108, it then forwards this shared, read-only copy to the requesting external device using a protocol that is appropriate for communicating with the requesting external device (generalized as the X-protocol XRTS_data command herein). Further, the new state of the tag that tracks this requested memory block is now upgraded to an eS state, signifying that an external device is currently caching a shared, read-only copy of this memory block.

[0095] If one of the external devices, e.g., I/O device 202, issues a read-to-share memory access request (using the X-protocol XRTS) for a given memory block and a shared, read-only copy of that memory block has already been cached by a sister external device, e.g., coherent domain device 204, there would already be a tag in tag array 250 for tracking this memory block. Further, the state of such tag will reflect an eS copy. In this case, there is no need to allocate a new tag to track the requested memory block.

[0096] In one embodiment, logic circuitry associated with coherence transformer 200 may obtain the shared, read-only copy of the requested memory block from the sister external device to satisfy the outstanding XRTO request In this embodiment, no action on common bus 108 is required. In another embodiment, coherence transformer 200 may obtain the requested shared, read-only copy of the requested memory block from the bus entities in computer system 100 by issuing, as shown in Fig. 11, a request-to-share (RTS) memory access request to common bus 108.

[0097] After coherence transformer 200 receives the shared, read-only copy of the requested memory block from common bus 108 (RTS_data transaction), coherence transformer 200 may then forward this copy to the requesting external device to service the XRTS memory access request (XRTS_data transaction). Further, the state associated with the matching tag in tag array 250 is maintained at an eS (shared) state.

[0098] If the memory access request received by coherence transformer 200 is a request for a shared, read-only copy of a memory block (XRTS) from an external device and a sister external device is currently caching the exclusive copy of that memory block, logic circuitry provided with coherence transformer 200 preferably obtains the requested memory block from the sister external device (and downgrades the previously existing exclusive copy) to satisfy the XRTS request without requiring the attention of coherence transformer 200 itself. On the other hand, if the XRTS memory access request comes from an external device that already is currently caching the exclusive copy of the requested memory block, an error condition exists as shown in Fig. 11. The error condition may be handle using a variety of conventional techniques, e.g., flag the error and/or perform a software or hardware reset.

A6. XWB Request

[0099] If the memory access request received by coherence transformer 200 is a write back transaction (X-protocol XWB transaction), i.e., an external device wishes to write back the exclusive copy of a memory block it currently owns, the actions of coherence transformer 200 depends on the state of the copy of the memory block currently cached by the external device. Generally, the external device that issues the write back transaction owns the exclusive copy of that memory block, and the current state of the matching tag in tag array 250 should show an eM (exclusive) state. Consequently, if the current state in the matching tag is eI (invalid) or eS (shared, read-only), an error condition exists as shown in Fig. 11. Again, this error condition may be handle using a variety of conventional techniques, including flagging the error and/or performing a software or hardware reset.

[0100] If the current state of the matching tag in tag array 250 is an eM (exclusive) state, coherence transformer 200 proceeds to receive the data to be written back (via the X-protocol XWB_data command) and issues a WB memory access request to common bus 108, to be followed up by the data (WB_data) (and also sets the Mtag to gM). Further, the external copy of the requested memory block is downgraded accordingly from an exclusive copy to an invalid copy (New State = eI).

[0101] To further clarify the details regarding the generalized X-protocol, which is employed by coherence protocol 200 in communicating with each external device, Tables 1 and 2 illustrate selected X-protocol requests and X-protocol responses.. It should be borne in mind that Tables 1 and 2 are shown for illustration purposes only and other requests and responses may also be provided depending on needs. As mentioned earlier, the adaptation of the disclosed generalized X-protocol transactions to work with a specific external coherence domain will depend greatly on the specification of the protocol employed by the specific external device and is generally within the skills of one skilled in the art.

[0102] In Table 1, the X-protocol requests, the actions represented by the requests, and possible responses thereto are shown. In Table 2, the X-protocol responses and the actions represented by the responses are shown. Table 2 further specifies whether a given response will be accompanied by data.

Table 1

REQUEST	ACTION	POSSIBLE RESPONSES
XRTO	Get exclusive copy of memory block	XRTO_data, XRTO_nack
XRTS	Get shared, read-only copy of memory block	XRTS_data, XRTS_nack
XINV	Invalidate copy of memory block	XINV_ack
XWB	Request to write back currently cached exclusive copy of memory block	XWB_ack, XWB_nack

Table 2

RESPONSES	ACTION	DATA?
XRTO_data	Reply with exclusive copy of memory block	Y
XRTO_nack	Not acknowledged, retry XRTO progenitor	N
XRTS_data	Reply with shared copy of memory block	Y
XRTS_nack	Not acknowledged, retry XRTS progenitor	N
XINV_ack	acknowledged	N
XWB_ack	acknowledged, permitting XWB_data	N
XWB_data	write back with exclusive copy of memory block	Y

[0103] Advantageously, the use of a coherence transformer and the tightly-coupled request-response transactions permit external devices, which may employ protocols different from the protocol on common bus 108 of computer system 100 to share memory blocks which have local physical addresses within computer system 100. Further, the explicit handshaking provided by the tightly coupled request-response pairs makes this sharing possible even if the external devices may each be operating at a different operating speed from that on common bus 108.

[0104] In a system in which coherence transformer 200 facilitates such memory block sharing, there is essentially no effect on system performance within computer system 100 when an external device does not cache a memory block. When an external device caches fewer memory blocks then there are tags in tag array 250 of coherence transformer 200, the effect on the overall system performance is fairly minimal. This is because when there are fewer externally-cached memory blocks than there are available tags in tag array 250, no additional transactions employing common bus 108, i.e., those associated with the Mtag-only approach to allow external devices to cache memory blocks without tracking them in snoop tag array 250.

[0105] The latency in responding to outstanding memory access requests on common bus 108 is due in part from the delay required for coherence transformer 200 to examine tags in tag array 250 to determine whether coherence transformer 200 should intervene to service a memory access request on common bus 108.

Section B: Mtag-only approach:

[0106] In the Mtag-only approach, i.e., the approach taken by steps 508 of Fig. 8 and 610 of Fig. 9), an externally originated memory access request can be serviced even though there is no room in snoop tag array 250 for tracking the externally cached memory block for the duration it is externally cached.

[0107] Figs. 12 and 13 show, in one embodiment of the Mtag-only approach, the memory access requests and responses issued by a bus entity, e.g., any of the entities coupled to common bus 108 such as processing units 102, 104, 106 or coherence transformer 200. In the description that follows, it is assumed for simplicity of illustration that there is only one bus entity internal to computer node 100, e.g., processing unit 102, being coupled to common bus 108. If there are more than one internal bus entities coupled to common bus 108, e.g., both processing units 102 and 104 are present on common bus 108, the resolution of coherence problems among these internal bus entities may be resolved using any conventional method.

[0108] By way of example, one solution to such coherence problems involves requiring each internal bus entity to snoop bus 108. If the snooped memory access request involves a memory block whose latest copy is cached by that internal bus entity, that internal bus entity may intervene to respond to the outstanding memory access request before memory module 110 may respond. An internal bus entity may ignore an outstanding memory access request if the request does not involve a memory block cached by that internal bus entity. If no internal bus entity intervenes, memory

module 110 is implicitly responsible for responding with the copy it currently possesses.

[0109] Referring now to Figs 12 and 13, a bus entity, e.g., processing node 102, may issue a memory access request for an exclusive copy of memory block 112(a) by issuing a request to own (RTO) request. In the description that follows, a request may have the form of request 400 of Fig. 5. On the other hand, a response may have the form of response 500 of Fig. 6.

[0110] If no other internal bus entities intervenes responsive to the RTO request, memory module 110 may respond to the outstanding RTO request with a RTO_data to furnish the RTO progenitor with a copy of the requested memory block from memory module 110, along with the state of that memory block (i.e., the content of the associated Mtag). Coherence transformer 200, as a bus entity, may not intervene if there is not a match between the requested memory block and one of the memory blocks tracked by tags in snoop tag array 250. If the RTO request is erroneous, e.g., requesting a non-existent memory block, memory module 110 may reply with a RTO-nack response, signifying that the RTO request is not acknowledged and needs to be retried by the RTO progenitor.

[0111] Once the RTO_data response is received by the RTO progenitor from memory block 110, i.e., by processing unit 102 in this example, the RTO progenitor then examines the state of the enclosed Mtag to determine whether the current copy of the memory block received from memory module 110 can be employed to service the issued RTO request. If the state is gI, for example, it is understood that an external device currently has the exclusive copy of the memory block, and the RTO progenitor may issue a request to obtain that copy and invalidate all external copies via the remote RTO memory access request (RRTO). Details regarding the RTO and RRTO requests, as well as other requests described herein, are discussed more fully herein, particularly with reference to Fig. 12.

[0112] If the Mtag state is gS, at least one external bus entity had a shared, read-only copy. In this case, it will be necessary to invalidate all shared copies existing internally and externally, and respond to the outstanding RTO request with the latest copy. If the state is gM, one of the internal entities has the latest valid copy and the RTO progenitor may proceed to employ the data returned in the RTO_data response from memory module 110 to satisfy its RTO needs (since it is assumed herein that there is no other internal entity to intervene with a later copy).

[0113] A remote RTO (RRTO) memory access request is typically issued by a RTO progenitor after that RTO progenitor finds out, by ascertaining the state of the Mtag received from memory module 110, that the state of the Mtag is insufficient to service the current RTO request. Insufficient Mtag states in this case may be gS or gI, i.e., there may be a shared or exclusive copy of the requested memory block existing externally. If the RRTO is issued by the RTO progenitor responsive to a gM Mtag, coherence transformer 200 understands this to be an error condition (since state gM indicates that the internal domain, not the external domain, currently has the exclusive copy of the requested memory block) and may request the RRTO progenitor to retry to obtain the exclusive copy from the internal domain.

[0114] If the RRTO is issued by the RTO progenitor responsive to a gS Mtag, coherence transformer 200 may respond to this RRTO command by invalidating external shared copy or copies, obtaining the latest copy of the requested memory block either from the external domain or the internal domain, invalidating all internal shared copy or copies, and returning that copy to the RRTO progenitor via the RTOR_data response. If the RRTO is issued by the RTO progenitor responsive to a gI Mtag, coherence transformer 200 may respond to this RRTO command by obtaining the external exclusive copy, invalidating that external exclusive copy, and returning that copy to the RRTO progenitor via the RTOR_data response. Further, coherence transformer 200 may perform a write back to memory module 110 to change the state of the Mtag corresponding to the requested memory block to gM via the RTOR response. If the RRTO request is erroneous, e.g., requesting a non-existent memory block, coherence transformer 200 may reply with a RTOR_nack response, signifying that the RRTO request is not acknowledged and needs to be retried by the RRTO progenitor.

[0115] A bus entity, e.g., processing node 102, may issue a memory access request for a shared, read-only copy of memory block 112(a) by issuing a RTS request. If no other internal bus entities intervenes, memory module 110 may respond to the outstanding RTS request with a RTS_data to furnish the RTS progenitor with a copy of the requested memory block from memory module 110, along with the state of that memory block (i.e., the content of the associated Mtag). Coherence transformer 200, as a bus entity, may not intervene if there is not a match between the requested memory block and one of the memory blocks tracked by tags in snoop tag array 250. If the RTS request is erroneous, e.g., requesting a non-existent memory block, memory module 110 may reply with a RTS-nack response, signifying that the RTS request is not acknowledged and needs to be retried by the RTS progenitor.

[0116] Once the RTS_data response is received by the RTS progenitor from memory block 110, i.e., processing unit 102 in this example, the RTS progenitor then examines the state of the enclosed Mtag to determine whether the current copy of the memory block received from memory module 110 can be employed to service the current RTS need. Generally, if the state of the Mtag is gS, at least one internal bus entity currently has a shared, read-only copy and this RTS memory access request can be serviced either by another internal bus entity or by the data received from memory module 110 itself. If the state of the Mtag is gM, at least one internal bus entity currently has an exclusive copy and this RTS memory access request can be serviced either by another internal bus entity or by the data received from memory module 110 itself.

[0117] If the state is gI, it is understood that an external device currently has the exclusive copy of the memory block and the RTS progenitor may issue a request to obtain that copy via the remote RTS memory access request (RRTS). If for some reason the RRTS is issued by the RTS progenitor responsive to a gM or gS Mtag, coherence transformer 200 understands this to be an error condition and will request the RTS progenitor to retry to obtain the shared copy from the internal bus entities. If the RRTS is issued by the RTS progenitor responsive to a gI Mtag, coherence transformer 200 may respond to this RRTS command by obtaining the shared copy of the requested memory block from the external device and returning that copy to the RRTS progenitor via the RTSR_data response. Further, coherence transformer 200 performs a write back to memory module 110 to change the state of the Mtag corresponding to the requested memory block to gS (via the RTSR response). If the RRTS request is erroneous, e.g., requesting a non-existent memory block, coherence transformer 200 may reply with a RTSR_nack response, signifying that the RRTS request is not acknowledged and needs to be retried by the RRTS progenitor.

[0118] Either one of the processing nodes, e.g., processing node 102, or coherence transformer 200 (on behalf of an external device) may issue a write back (WB) request to write back to memory 110 an exclusive copy of a memory block it earlier cached. If the WB request is erroneous, e.g., requesting a non-existent memory block, memory module 110 may reply with a WB_nack response, signifying that the WB request is not acknowledged and needs to be retried by the WB progenitor.

[0119] On the other hand, if no WB_nack response is issued, the WB progenitor may follow up with a WB_data response to write back the memory block to memory module 110. Further, the state of the Mtag in memory module 110 may also be changed to gM (if coherence transformer 200 requests the write back) to reflect the fact that the internal domain now has the exclusive copy of this memory block.

[0120] As mentioned earlier, when there is a remote memory access request, e.g., an RRTO or a RRTS, on common bus 108, coherence transformer 200 (via coherence transformer link 220) receives this memory access request and formulates an appropriate response depending on the state of the Mtag. The operation of the coherence transformer 200 may be more clearly understood with reference to Figs. 14 and 15.

[0121] Fig. 14 illustrates, in one embodiment of the present invention, selected transactions performed by coherence transformer 200 in response to remote memory access requests on common bus 108. Referring now to Fig. 14, when a remote memory access request is issued by one of the internal bus entities on common bus 108, this remote memory access request is forwarded to all bus entities, including coherence transformer 200. The remote request may be, however, ignored by all internal bus entities, e.g., processor 102. Responsive to the remote request, coherence transformer 200 ascertains the current state of the Mtag (included in the remote request) to determine whether one of the external devices has an appropriate copy of the requested memory block for responding to the remote memory access request on common bus 108. The ascertaining of the Mtag state is necessary since it has been determined, in step 504 of Fig. 8, that there is no match between the requested memory block and one of the memory blocks tracked in snoop tag array 250.

B1. Remote Request to Own (RRTO)

[0122] If the remote memory access request is a request for an exclusive copy of a memory block (a RRTO) and the current Mtag state is gM, coherence transformer 200 understands this to be an error condition (since state gM indicates that the internal domain, not the external domain, currently has the exclusive copy of the requested memory block) and may request the RRTO progenitor to retry to obtain the exclusive copy from the internal domain.

[0123] On the other hand, the RRTO may be issued by the RTO progenitor in response to a gS or a gI Mtag. This occurs when the external domain is currently caching a valid copy of the requested memory block and there is no room in snoop tag array 250 for tracking this externally cached memory block. Consequently, coherence transformer 200 would not be able to intervene to respond when the original RTO (issued by the RTO progenitor pertaining to this memory block) was present on common bus 108. With reference to Fig.8, coherence transformer 200 finds no tag match in step 504 and proceeds to let memory module 110 respond (in step 508 of Fig. 8 in accordance with the Mtag-only approach).

[0124] If the RRTO is issued by the RTO progenitor responsive to a gS Mtag, coherence transformer 200 may respond to this RRTO command by invalidating external shared copy or copies by issuing the X-protocol invalidate command XINV to request all external devices to invalidate their shared copies. Coherence transformer 200 may either broadcast the X-protocol commands or may simply direct the X-protocol command to the appropriate external device(s) if there is provided logic with coherence transformer 200 for keeping track of the locations and types of memory blocks cached.

[0125] When all external copies have been invalidated (confirmed by the receipt of the X-protocol XINV_ack response) coherence transformer 200 may then obtain the latest copy of the requested memory block from the internal domain and invalidate all internal shared copy or copies. In one embodiment, coherence transformer 200 may obtain the latest copy of the requested memory block from the internal domain and invalidate all internal shared copy or copies by issuing a RTO request to common bus 108. Upon receiving the requested copy from the internal domain (via the

RTO_data response), coherence transformer 200 may write back the copy to memory module 110 along with the appropriate Mtag, i.e., gM in this case, via the RTOR response. Thereafter, coherence transformer 200 may provide the requested copy to the RRTO progenitor via the RTOR_data response.

[0126] Note that the use of the XINV command advantageously invalidates all shared copies of the requested memory block cached by the external device(s). Further, the use of the RTO request by coherence transformer 200 to common bus 108 advantageously ensures that all internal shared copies within computer node 100 is invalidated and obtains the required memory block copy to forward to the requesting entity, i.e., the RRTO progenitor.

[0127] If the RRTO request is issued by the RTO progenitor responsive to a gI Mtag, coherence transformer 200 may respond to this RRTO command by obtaining the external exclusive copy and invalidating that external exclusive copy via the X-protocol XRTO request. When the external exclusive copy is obtained (via the X-protocol XRTO_data response), coherence transformer 200 may perform a write back to memory module 110 to change the state of the Mtag corresponding to the requested memory block to gM via the RTOR response. Further, coherence transformer 200 may return the copy of the requested memory block to the RRTO progenitor via the RTOR_data response.

B2. Remote Request to Share (RRTS)

[0128] If the remote memory access request is a request for a shared copy of a memory block (a RRTS) and the current state of the Mtag is gM or gS, coherence transformer 200 understands this to be an error condition (since these states indicate that there is at least one valid, i.e., shared or exclusive, copy internally) and will request the RTS progenitor to retry to obtain the shared copy from the internal bus entities. If the RRTS is issued by the RTS progenitor responsive to a gI Mtag, coherence transformer 200 may respond to this RRTS command by obtaining the shared copy of the requested memory block from the external device (via the X-protocol XRTS request). When the external shared copy is obtained (via the X-protocol XRTS_data response), coherence transformer 200 may perform a write back to memory module 110 to change the state of the Mtag corresponding to the requested memory block to gS via the RTSR response. Further, coherence transformer 200 may return the copy of the obtained memory block to the RRTS progenitor via the RTSR_data response.

[0129] Coherence transformer 200 not only interacts with the processing nodes within computer nodes 100 to respond to remote memory access requests issued by those processing nodes, it also interacts with the external devices, e.g., external devices 202, 204, and 206, in order to service memory access requests pertaining to memory blocks having local physical addresses within computer node 100.

[0130] Fig. 15 illustrates selected transactions performed by coherence transformer 200 in response to memory access requests from one of the external devices. In Fig. 15, the memory access requests are issued, using the aforementioned generalized X-protocol, by one of the external devices, e.g., one of devices 202, 204, or 206, to coherence transformer 200. If another external device currently caches the required copy of the requested memory block, this memory access request is preferably handled by logic circuitry provided with coherence transformer 200 without requiring the attention of coherence transformer 200 itself.

[0131] On the other hand, if another external device does not have the valid copy of the requested memory block to service the memory access request, coherence transformer 200 then causes a memory access request to appear on common bus 108, using a protocol appropriate to computer node 100, so that coherence transformer 200 can obtain the required copy of the requested memory block on behalf of the requesting external device. Further, since a copy of the memory block is now cached by an external device, and it has been determined in step 606 of Fig. 9 that there is no additional room in snoop tag array 250 to track this externally requested memory block for the entire duration of it being externally cached, the Mtag associated with this memory block may need to be changed in memory module 110 to reflect this change, i.e., the invention proceeds to handle this externally originated memory access request using the Mtag-only approach.

B3. XRTO Memory Access Request

[0132] Referring now to Fig. 15, when an external device issues a memory access request to obtain an exclusive copy of a memory block having a local physical address within computer node 100, e.g., memory block 112(a), it issues a XRTO request to coherence transformer 200. Coherence transformer 200 then obtains the copy of the requested memory block from the internal domain and invalidates all internal copies of the request memory block (by issuing a RTO request to common bus 108). After receiving the copy of the requested memory block, coherence transformer 200 then ascertains the state of the associated Mtag to determine its next course of action.

[0133] If the state of the Mtag (contained in the RTO_data response) is gI, coherence transformer 200 understands this to be an error since the external domain does not have the exclusive copy (otherwise it would not need to request the exclusive copy from the internal domain) and the internal domain does not have either a shared or exclusive copy (gI Mtag state). The error condition may be handled using a variety of conventional techniques, e.g., flag the error and/

or perform a software or hardware reset.

[0134] On the other hand, if the state of the Mtag is gM or gS, coherence transformer 200 then writes back to memory module 110 (via the WB request and WB_data response) the new state, i.e., gI, to signify that there is no longer a valid copy of the requested memory block in the internal domain. In one embodiment, the write back may be performed with only the new state gI and without any other data for the requested memory block to save bandwidth on common bus 108 (since any data associated with an invalid Mtag state would be ignored anyway). Thereafter, coherence transformer 200 may forward the copy of the obtained memory block to the requesting external device via the X-protocol XRTO_data response.

B4. XRTS Memory Access Request

[0135] When an external device issues a memory access request to obtain a shared copy of a memory block having a local physical address within computer node 100, e.g., memory block 112(a), it issues a XRTS request to coherence transformer 200. Coherence transformer 200 then obtains the copy of the requested memory block from the internal domain and writes the gS state to memory module 110 (by issuing a RTSM request to common bus 108 and receives the RTSM_data response). If the state of the Mtag is gI, coherence transformer 200 typically would receive a response from the memory module with Mtag gI. If the response is received and the Mtag state contained in the RTSM_data response is gI or, for some reason, there is no response, coherence transformer 200 understands this to be an error since the external domain does not have the exclusive copy (otherwise it would not need to request the exclusive copy from the internal domain) and the internal domain does not have either a shared or exclusive copy (gI Mtag state). The error condition may be handled using a variety of conventional techniques, e.g., flag the error and/or perform a software or hardware reset.

[0136] On the other hand, if the state of the Mtag is gM or gS, coherence transformer 200 may forward the copy of the obtained memory block to the requesting external device via the X-protocol XRTS_data response.

[0137] Note that the RTSM and RTSM_data sequence may equally be substituted by a sequence containing RTO (from coherence transformer 200 to common bus 108), RTO_data (from common bus 108 to coherence transformer 200), WB (from coherence transformer 200 to common bus 108 to ask permission to write to memory module 110), and WB_data (writing the gS Mtag to the corresponding memory block in memory module 110.

B5. XWB Request

[0138] When an external device issues a request to write back an exclusive copy of a memory block it earlier cached from computer node 100, it issues a X-protocol XWB request to coherence transformer 200. In accordance with the Mtag-only approach, coherence transformer 200 may then obtain a copy of the requested memory block from the internal domain to ascertain the current state of the associated Mtag. If the current state is gM or gS, coherence transformer 200 understands this to be an error since the external domain, which requests to write back, must have the only valid, exclusive copy and there must be no other valid (whether exclusive or shared) copy of the same memory block anywhere else in the computer system. The error condition may be handled using a variety of conventional techniques, e.g., flag the error and/or perform a software or hardware reset.

[0139] On the other hand, if the state of the Mtag is gI, coherence transformer 200 then proceeds to receive from the external device the data to be written back (via the X-protocol XWB_data response) and writes this data, along with the new gM Mtag state, to the appropriate memory location in memory module 110. In one embodiment, the writing of both the data and the gM Mtag state can be accomplished by issuing a WSgM command to common bus 108, which requests the writing of both data and new Mtag, to be followed by the data and the new gM Mtag in the WSgM_data command.

[0140] Note that the WSgM and WSgM_data sequence may well be substituted by a sequence containing RTO (from common bus 108 on behalf of memory module 110 to coherence transformer 200), RTO_data (from coherence transformer 200 to common bus 108 to furnish the old data be overwritten from memory module 110), WB (from coherence transformer 200 to common bus 108 to ask permission to write to memory module 110), and WB_data (writing the gM Mtag to the corresponding memory block in memory module 110).

[0141] In accordance with the Mtag-only approach, the use of a coherence transformer and the tightly-coupled request-response transactions, advantageously permit external devices, which may be employing protocols different from the protocol on common bus 108 of computer node 100, to share memory blocks having local physical addresses within computer node 100. Further, coherence transformer 200 makes this sharing possible even if the external devices may each be operating at a different operating speed from that on common bus 108.

[0142] Note that the external devices do not need to accommodate Mtags to participate in memory sharing. Only the bus entities, e.g., memory module 110, the processors coupled to common bus 108, and coherence transformer 200, need to be aware of the existence of Mtags to employ them in avoiding coherence problems. Consequently, this

feature of coherence transformer 200 advantageously permits a properly configured computer node 100 to work with a wide range of existing external devices to facilitate memory sharing without requiring any modification to the external devices.

[0143] The Mtag-only approach advantageously permits the external devices to cache any number of memory blocks. Due to the existence of Mtags, coherence transformer 200 advantageously does not need to keep track of every memory block currently cached by the external devices for the purpose of deciding whether coherence transformer 200 should intervene in servicing a memory access request on common bus 108. This is in contrast with the snoop-only approach, which requires a tag to track every externally cached memory block and which is employed herein only when there is still room in snoop tag array 250 to track the externally cached memory blocks. When there is no more room in snoop tag array 250 to track the externally cached memory blocks, the Mtag-only approach can advantageously be employed to facilitate the caching of memory blocks by external devices in a coherent manner.

[0144] In accordance with one aspect of the inventive Mtag-only approach, the bus entity that obtains the memory block from memory module 110 decides for itself, upon ascertaining the Mtag state of the obtained memory block, whether it needs to further request a more recent copy from the external device (via the remote requests RRTO and RRTS directed at coherence transformer 200).

[0145] In one embodiment, coherence transformer 200 in the Mtag-only approach is provided with at least one buffer block, e.g., buffer 280 of Fig. 7A, for temporarily storing a copy of the memory block most recently accessed by one of the external device. The buffer block may store both the address of the memory block and the relevant Mtag data (or alternatively the external states eS, eI, or eM since Mtags and external states can be derived from one another). The buffer block advantageously permits coherence transformer 200 to perform write back to memory module 110 to change the state of the Mtag in memory module 110.

[0146] While operating in the Mtag-only approach, in the interval after coherence transformer 200 obtains the copy of the memory block requested and before coherence transformer 200 performs a write back to change the Mtag, e. g., responsive to a XRTO request from an external device, coherence transformer 200 may, using the data stored in the buffer, monitor common bus 108 to intervene. The intervention may be necessary if, for example, another internal bus entity requests this memory block during the aforementioned interval.

[0147] Note that once the write back is performed to change the Mtag to the appropriate state, it is no longer necessary to keep a copy of that memory block in the buffer. Because a copy of a memory block is typically kept in a buffer for a very short time, the number of buffers required may be quite small.

[0148] Further, since a response to an externally-originated memory access request, e.g., XRTO, XRTS or XWB, requires knowledge of the state of the corresponding Mtag, there is optionally provided, as an optimization technique in one embodiment, a Mtag cache array for tracking some or all memory blocks of memory module 110. For example, a Mtag cache array may be provided to track only the Mtag states of the memory blocks externally cached. Alternatively, a Mtag cache array may be employed to track the Mtag states of every memory block in memory module 110.

[0149] As another embodiment, an Mtag cache array may be provided to track only memory blocks whose Mtag states are gS and gI. This embodiment is particularly advantageous in computer systems in which a relatively small number of memory blocks are externally cached at any given time. In such a computer system, most memory blocks would have a gM state, and relative few would have gS and gI Mtag states

[0150] When coherence transformer 100 requires knowledge of the Mtag state associated with a given memory block, it checks the Mtag cache array first. In case of a cache hit, no bandwidth of common bus 108 is required to ascertain the Mtag state. In case of the cache miss, coherence transformer 200 may proceed to inquire, via common bus 108 as discussed herein, the state of the associated Mtag to determine its proper course of action. Note that the presence of a Mtag cache array is not absolutely necessary and is equally well to have an implementation wherein no Mtag caching is performed (in which case coherence transformer inquires, via common bus 108, the Mtag state when it needs this information).

[0151] As is apparent from the foregoing, when there is sufficient room in snoop tag array 250 of snoop coherence transformer 200 to keep track of externally cached memory blocks, the embodiment advantageously operates in the snoop-only mode. In this mode; the performance of the hybrid approach described herein is as efficient as that of the snoop techniques. Note that in the snoop-only approach, when an external device caches a memory block, there is advantageously no need to write back to memory module 110 the new Mtag. In this manner, common bus 108 is only employed once to furnish the requested memory block to coherence transformer 200 to service the externally originated memory access request, and there is no need to use common bus 108 again to write the new Mtag to memory module 110, as would be required in the Mtag-only approach.

[0152] When there is no room left in snoop tag array 250 to keep track of externally cached memory blocks, the embodiment advantageously switches to the Mtag-only approach. In this approach, there is no need to track the externally cached memory block in snoop tag array 250.

[0153] Although additional bandwidth on common bus 108 is required to write back the new Mtag state to memory module 110, the Mtag-only approach is still an advantageous mode of operation when there is no room left in snoop

tag array 250. This is because the Mtag-only approach, unlike the snoop approach, does not require the forcible write back of a memory block that is externally cached previously for the purpose of freeing up a tag in snoop tag array 250 in order to track the newly cached memory block. The forcible write back of a memory block that is externally cached earlier is a time consuming operation since snoop coherence transformer 200 must decide which of the multiple memory blocks externally cached should be written back, and must go out to the external device to invalidate the externally cached copy before writing it back to memory module 110. Such an action is required in the snoop-only approach whenever there is no more tag in snoop tag array 250 since the snoop only approach requires that each externally cached memory block be tracked by a tag in snoop tag array 250, and unless a forcible write back is performed on a memory block that is externally cached previously to unallocate a tag, there is no tag in snoop tag array 250 to service the new externally requested memory block.

[0154] As is apparent, the exact number of tags in snoop tag array 250 depends on the needs of a particular system. In general, as many tags as reasonably possible should be provided in snoop tag array 250 to defer the need to operate in the Mtag-only approach. Note that the tags are recycled since when an external device writes back a memory block that is tracked in snoop tag array 250, the tag that is employed to track this externally cached memory block is then freed up, allowing coherence transformer 200 to service the next externally originated memory access request using the snoop-only approach.

[0155] Note that the embodiment, with a finite number of tags in snoop tag array 250, takes advantage of the optimum operating range of the snoop approach while avoiding its less efficient operating point. The embodiment advantageously operates in the optimum operating range of the snoop approach when there are tags in snoop tag array 250 to track externally cached memory blocks, thereby avoiding the inefficiency associated with the Mtag-only approach in this operating range (i.e., the need of the Mtag approach to write the Mtag back to memory module 110).

[0156] When there are no more tags in snoop tag array 250 to track externally cached memory blocks, the embodiment advantageously avoids the more inefficient operating mode associated with the snoop approach for this operating range (i.e., the mode wherein forcible write backs of memory blocks externally cached in previous cycles are necessitated). In this operating range, the embodiment advantageously switches to an Mtag-only operating mode, thereby allowing the system to operate at a relatively higher efficiency.

[0157] The embodiment has been described as allowing one coherence transformer per bus. System designers may, in some cases, want to attach several coherence transformers to a bus to connect many alternative device of the same or different types, e.g. I/O devices, DSM memory agents, coherence domain devices, and the like. The implementation of multiple coherence transformers would be apparent to those skilled in the art given this disclosure. In a multiple coherence transformer implementation, Mtags may be extended with a field to identify which coherence transformer has the block externally so that processors know which coherence transformer should receive the appropriate RRTO's and RRTS's.

[0158] It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention as claimed. By way of example, some systems may crate an illusion of a common bus without requiring a physical bus (e.g., via a set of broadcast wires). The KSR-1 from Kendall Square Research of Massachusetts is one such example. The present invention applies equally well to these and other analogous systems.

Claims

1. A method of enabling an external device (202, 204, 206) in an external domain that is external to a computer node (100) of a computer system to share memory blocks (112) having local physical addresses in a memory module (110) at said computer node irrespective whether said external device and a common bus (108) at said computer node both employ a common protocol and irrespective whether said external device and said common bus both operate at the same speed, said computer node including a coherence transformer (200), said memory module and a processing node connected to said common bus, said processing node (102, 104, 106) having a processor (116) and a cache (114), each of said memory blocks having an associated Mtag for tracking a global state associated with each memory block, including a global exclusive state for indicating that each memory block is exclusive to said computer node, a global shared state for indicating that each memory block is shared by said computer node with said external device, and a global invalid state for indicating that each memory block is invalid in said computer node, said method comprising:

   snooping said common bus to monitor memory access requests on said common bus;

   receiving, at the coherence transformer, a first memory access request for caching a first memory block from said external device;

obtaining a first copy of said first memory block, using said coherence transformer, from said common bus, **characterised in that** said coherence transformer having a snoop tag array (250) having a plurality of snoop tags, each of said plurality of snoop tags being configured to identify one of said memory blocks if cached by said external device and to track an external state of a copy of that memory block, said external state including one of an external exclusive state for indicating that said copy of that memory block is exclusive to said external domain, an external shared state for indicating that said copy of that memory block is shared by said external domain, and an external invalid state for indicating that said copy of that memory block is invalid in said external domain; and

if at least one tag in said plurality of snoop tags is available for tracking said external state of said first copy of said first memory block, responding to said first memory access request using a snoop-only approach in which that tag is used to track said external state of said first copy of said first memory block for an entire duration, that said first memory block is cached by said external device;

else if at least one tag in said plurality of snoop tags is not available for tracking said external state of said first copy of said first memory block, responding to said first memory access request using an Mtag-only approach in which, using said coherence transformer, a tag for said first memory block is temporarily stored until a global state associated with said first memory block can be written back into said memory module;

said first copy of said first memory block being sent from said coherence transformer to said external device.

2. The method of claim 1 wherein said first memory access request from said external device represents a request for an exclusive copy of said first memory block and said step of responding to said first memory access request using said Mtag-only approach further includes a step of changing said first Mtag in said memory module to a global invalid state.

3. The method of claim 2 wherein said step of responding to said first memory access request using said Mtag-only approach further comprising a step of invalidating all valid copies of said first memory block at said computer node.

4. The method of claim 1 wherein said first memory access request from said external device represents either a request for an exclusive copy of said first memory block or a request for a shared copy of said first memory block, said step of responding to said first memory access request using said Mtag-only approach further includes the steps of:

prior to said modifying step, examining said first Mtag associated with said first memory block; and

proceeding with said modifying step and said sending step only if said first Mtag does not represent a global invalid state.

5. The method of claim 1 wherein said first memory access request from said external device represents a request for a shared copy of said first memory block and said step of responding to said first memory access request using said Mtag-only approach further includes a step of changing said first Mtag in said memory module to a global shared state.

6. The method of claim 5 wherein said step of responding to said first memory access request using said Mtag-only approach further includes the steps of:

prior to said modifying step, examining said first Mtag associated with said first memory block;

proceeding with said modifying step and said sending step only if said first Mtag does not represent a global invalid state; and

if said first Mtag represents a global invalid state, flagging an error condition.

7. The method of claim 1 further comprising the steps of:

receiving a write back request for a second memory block from said external device at said coherence transformer;

obtaining said first copy of said second memory block, using said coherence transformer, from said external device;

writing said first copy of said second memory block from said coherence transformer to said memory module at said computer node; and

if said first copy of said first memory block is not tracked in a snoop tag of said snoop tag array, modifying, using said coherence transformer, an Mtag associated with said second memory block in said memory module at said computer node to reflect that said computer node has an exclusive copy of said second memory block.

8. The method of claim 1 wherein said global state for said each of said memory blocks is employed as said external state for said each of said memory blocks, whereby a global exclusive state represents an external invalid state, a global shared state represents an external shared state, and a global invalid state represents an external exclusive state.

9. The method of claim 1 further comprising the steps of:

receiving a writeback request for said first memory block from said external device at said coherence transformer,

obtaining said first copy of said first memory block, using said coherence transformer, from said external device;

writing said first copy of said first memory block from said coherence transformer to said memory module at said computer node; and

if said first copy of said first memory block was tracked in a snoop tag of said snoop tag array prior to said writing step, unallocating said snoop tag of said snoop tag array, thereby rendering said snoop tag available for tracking other externally cached memory blocks and causing said first copy of said first memory block to be no longer tracked by said snoop tag array.

10. The method of claim 1 further comprising the step of responding, through said coherence transformer, to a second memory access request on said common bus on behalf of said external device, comprising:

monitoring memory access requests on said common bus, using said coherence transformer, to determine whether a second memory access request of said memory access requests on said common bus pertains to any one of memory blocks tracked in snoop tags of said snoop tag array; and

if said second memory access request pertains to a second memory block, said second memory block representing said one of memory blocks tracked in said snoop tags of said snoop tag array ,responding to said second memory access request using said snoop-only approach, including responding to said second memory access request using said coherence transformer.

11. The method of claim 10 wherein said coherence transformer responds to said second memory access request in said snoop-only approach only if a snoop tag tracking said second memory block in said snoop tag array indicates that a first copy of said second memory block is valid at said external device.

12. The method of claim 11 wherein said second memory access request is a request for an exclusive copy and said snoop tag tracking said second memory block indicates that said first copy of said second memory block at said external device is an exclusive copy of said second memory block, said step of responding to said second memory access request using said snoop-only approach comprises:

obtaining, using said coherence transformer, a second copy of said second memory block from said first copy of said second memory block at said external device;

invalidating said first copy of said second memory block at said external device; and

forwarding said second copy of said second memory block from said coherence transformer to said common bus to enable a progenitor of said second memory access request to obtain said second copy of said second

memory block; and

unallocating said snoop tag of said snoop tag array, thereby rendering said snoop tag available for tracking other externally cached memory blocks and causing said second memory block to be no longer tracked by said snoop tag array.

13. The method of claim 11 wherein said second memory access request is a request for an exclusive copy and said snoop tag tracking said second memory block indicates that said first copy of said second memory block at said external device is a shared copy of said second memory block, said step of responding to said second memory access request using said snoop-only approach comprises:

invalidating said first copy of said second memory block at said external device;

obtaining, using said coherence transformer, a second copy of said second memory block from said computer node via said common bus;

invalidating, using said coherence transformer, any valid copy of said second memory block in said computer node; and

forwarding said second copy of said second memory block from said coherence transformer to said common bus to enable a progenitor of said second memory access request to obtain said second copy of said second memory block; and

unallocating said snoop tag of said snoop tag array, thereby rendering said snoop tag available for tracking other externally cached memory blocks and causing said second memory block to be no longer tracked by said snoop tag array.

14. The method of claim 11 wherein said second memory access request is a request for a shared copy and said snoop tag tracking said second memory block indicates that said first copy of said second memory block at said external device is a shared copy of said second memory block, said step of responding to said second memory access request using said snoop-only approach comprises:

obtaining, using said coherence transformer, a second copy of said second memory block from said computer node via said common bus; and

forwarding said second copy of said second memory block from said coherence transformer to said common bus to enable a progenitor of said second memory access request to obtain said second copy of said second memory block.

15. The method of claim 11 wherein said second memory access request is a request for a shared copy and said snoop tag tracking said second memory block indicates that said first copy of said second memory block at said external device is an exclusive copy of said second memory block, said step of responding to said second memory access request using said snoop-only approach comprises:

obtaining, using said coherence transformer, a second copy of said second memory block from said external device;

changing said snoop tag tracking said second memory block to indicate that said first copy of said memory block at said external device is a shared copy of said second memory block; and

forwarding said second copy of said second memory block from said coherence transformer to said common bus to enable a progenitor of said second memory access request to obtain said second copy of said second memory block.

16. A coherence transformer (200) for facilitating the sharing of memory blocks (112) between a computer node (100) and an external device, said computer node including a common bus (108) to which said coherency transformer, a memory module (110) and a processing node (102, 104, 106) with a processor and cache (114) are connected, said memory blocks having local physical addresses in the memory module at said computer node, each of said

memory blocks having an associated Mtag for tracking a global state associated with each memory block, including a global exclusive state for indicating that memory block is exclusive to said computer node, a global shared state for indicating that memory block is shared by said computer node with said external device, and a global invalid state for indicating that each memory block is invalid in said computer node, said coherence transformer comprising:

snooping logic (260) configured for coupling with the common bus of said computer node, said snooping logic, when coupled to said common bus, being operable to monitor memory access requests on said common bus; said coherence transformer being **characterised by** comprising:

a snoop tag array (250) coupled to said snooping logic, said snoop tag array having a plurality of snoop tags (273,274,276,278,280), each of said plurality of snoop tags being configured to identify one of said memory blocks if cached by said external device and to track an external state of a copy of that memory block, said external state including one of an external exclusive state for indicating that said copy of that memory block is exclusive to said external domain, an external shared state for indicating that said copy of that memory block is shared by the said external domain, and an external invalid state for indicating that said copy of that memory block is invalid in said external domain; and

logic means for ascertaining (504, 506, 508) whether a first memory access request from said external device for caching a first memory block should be responded to using a snoop-only approach in which a tag in said snoop tag array is operable to track said external state of a copy of said first memory block for an entire duration that said first memory block is cached by said external device, or using an Mtag-only approach in which a tag for said first memory block is temporarily stored until a global state associated with said first memory block can be written back into said memory module.

17. The coherence transformer of Claim 16 further comprising logic (606,608,610) for ascertaining whether a second memory access for a second memory block on said common bus should be responded to using said snoop-only approach or said Mtag-only approach, said second memory access being responded to by said coherence transformer using said snoop-only approach when said second memory block is tracked by said snoop tag array, said second memory block being responded to by said memory module using said Mtag-only approach when said second memory access is not tracked by said snoop tag array.

18. A computer system having a computer node (100), said coherence transformer (200) of claim 16 or claim 17 and an external device, said computer node including a common bus (108) to which said coherence transformer (200), a memory module (110) and a processing node (102, 104, 106) with a processor (116) and a cache (114) are connected.

Patentansprüche

1. Verfahren zum Zulassen, dass eine externe Einrichtung (202, 204, 206) in einem externen Domain, das außerhalb eines Computerknotens (100) eines Computersystems liegt, Speicherblöcke (112) mit lokalen physikalischen Adressen in einem Speichermodul (110) an dem Computerknoten gemeinsam zu benutzen unabhängig davon, ob die externe Einrichtung und ein gemeinsamer Bus (108) an dem Computerknoten beide ein gemeinsames Protokoll verwenden und unabhängig davon, ob die externe Einrichtung und der gemeinsame Bus beide mit derselben Geschwindigkeit arbeiten, wobei der Computerknoten einen Kohärenztransformator (200) einschließt, das Speichermodul und einen mit dem gemeinsamen Bus verbundenen Verarbeitungsknoten, wobei der Verarbeitungsknoten (102, 104, 106) einen Prozessor (116) hat und einen Cache-Speicher (114), jeder der Speicherblöcke ein zugeordnetes Speicheretikett bzw. Mtag hat zum Verfolgen eines globalen Zustandes, der jedem Speicherblock zugeordnet ist einschließlich eines globalen Ausschließlichkeits-Zustandes zum Anzeigen, dass der jeweilige Speicherblock ausschließlich für den Computerknoten ist, eines globalen Geteilt-Zustandes zum Anzeigen, dass der jeweilige Speicherblock geteilt wird durch den Computerknoten mit externen Einrichtungen und eines globalen Ungültigkeits-Zustandes zum Anzeigen, dass der jeweilige Speicherblock ungültig ist in dem Computerknoten, wobei das Verfahren umfasst:

Beschnüffeln des gemeinsamen Busses zum Überwachen der Speicherzugriffsanforderungen auf dem gemeinsamen Bus; Empfangen einer ersten Speicherzugriffsanforderung beim Kohärenztransformator zum Cachen eines ersten Speicherblocks von der externen Einrichtung;

Erhalten einer ersten Kopie von dem ersten Speicherblock unter Verwendung des Kohärenztransformators von dem gemeinsamen Bus, ***dadurch gekennzeichnet,*** **dass** der Kohärenztransformator ein Schnüffeletiketten-Array bzw. Schnüffel-Tag-Array (250) hat mit einer Vielzahl von Schnüffel-Tags, jedes der Vielzahl von Schnüffel-Tags konfiguriert ist zum Identifizieren eines der Speicherblöcke, wenn er von der externen Einrichtung gecached ist und zum Verfolgen eines externen Zustandes einer Kopie des Speicherblocks, wobei der externe Zustand eines einschließt von einem externen Ausschließlichkeits-Zustand zum Anzeigen, dass die Kopie des Speicherblocks ausschließlich für die externe Domain ist, eines externen Geteilt-Zustandes zum Anzeigen, dass die Kopie des Speicherblocks geteilt wird durch die externe Domain, und eines externen Ungültigkeits-Zustandes zum Anzeigen, dass die Kopie des Speicherblocks ungültig ist in der externen Domain; und

wenn mindestens ein Tag in der Vielzahl von Schnüffel-Tags verfügbar ist zum Verfolgen des externen Zustandes der ersten Kopie des ersten Speicherblocks, ansprechend auf die erste Speicherzugriffanforderung unter Verwendung einer Nur-Schnüffel-Methode, bei der das Tag verwendet wird zum Verfolgen des externen Zustandes der ersten Kopie des ersten Speicherblocks für eine gesamte Dauer, zu der der erste Speicherblock gecached wird durch die externe Einrichtung;

andernfalls, wenn mindestens ein Tag in der Vielzahl von Schnüffel-Tags nicht verfügbar ist zum Verfolgen des externen Zustandes der ersten Kopie des ersten Speicherblocks ansprechend auf die erste Speicherzugriffsanforderung unter Verwendung einer Nur-Mtag-Methode, bei der unter Verwendung des Kohärenztransformators ein Tag für den ersten Speicherblock temporär gespeichert wird, bis ein globaler Zustand, der dem ersten Speicherblock zugeordnet ist, zurückgeschrieben werden kann in das Speichermodul;

wobei die erste Kopie des ersten Speicherblocks von dem Kohärenztransformator zu der externen Einrichtung gesendet wird.

2. Verfahren nach Anspruch 1, wobei die erste Speicherzugriffsanforderung von der externen Einrichtung eine Anforderung repräsentiert für eine exklusive Kopie des ersten Speicherblocks und der Schritt des Ansprechens auf die erste Speicherzugriffsanforderung unter Verwendung der Nur-Mtag-Methode außerdem einen Schritt einschließt des Änderns des ersten Mtag in dem Speichermodul zu einem globalen Ungültigkeits-Zustand.

3. Verfahren nach Anspruch 2, wobei der Schritt des Ansprechens auf die erste Speicherzugriffsanforderung unter Verwendung der Nur-Mtag-Methode außerdem einen Schritt umfasst des Ungültigmachens aller gültigen Kopien des ersten Speicherblocks bei dem Computerknoten.

4. Verfahren nach Anspruch 1, wobei die erste Speicherzugriffsanforderung von der externen Einrichtung entweder eine Anforderung bezüglich einer ausschließlichen Kopie des ersten Speicherblocks repräsentiert oder eine Anforderung bezüglich einer geteilten Kopie des ersten Speicherblocks, wobei der Schritt des Ansprechens auf die erste Speicherzugriffsanforderung unter Verwendung der Nur-Mtag-Methode außerdem die Schritte umfasst:

vor dem Modifizierungsschritt, Prüfen des ersten, dem ersten Speicherblock zugeordneten Mtag; und

Fortfahren mit dem Modifizierungsschritt und dem Sendeschritt nur, wenn das erste Mtag nicht einen globalen Ungültigkeits-Zustand repräsentiert.

5. Verfahren nach Anspruch 1, wobei die erste Speicherzugriffsanforderung von der externen Einrichtung eine Anforderung für eine geteilte Kopie des ersten Speicherblocks repräsentiert und der Schritt des Ansprechens auf die erste Speicherzugriffsanforderung unter Verwendung der Nur-Mtag-Methode außerdem einen Schritt einschließt des Änderns des ersten Mtags in dem Speichermodul zu einem globalen Geteilt-Zustand.

6. Verfahren nach Anspruch 5, wobei der Schritt des Ansprechens auf die erste Speicherzugriffsanforderung unter Verwendung der Nur-Mtag-Methode außerdem die Schritte einschließt:

vor dem Modifizierungsschritt, Prüfen des ersten Mtag, das dem ersten Speicherblock zugeordnet ist;

Fortschreiten mit dem Modifizierungsschritt und dem Sendeschritt nur, wenn das erste Mtag nicht einen globalen Ungültigkeits-Zustand repräsentiert; und

wenn das erste Mtag einen globalen Ungültigkeits-Zustand repräsentiert, Anzeigen einer Fehlerbedingung.

7. Verfahren nach Anspruch 1, außerdem die Schritte umfassend:

Empfangen einer Rückschreibanforderung für einen zweiten Speicherblock von der externen Einrichtung bei dem Kohärenztransformator:

Erhalten der ersten Kopie des zweiten Speicherblocks unter Verwendung des Kohärenztransformators von der externen Einrichtung;

Schreiben der ersten Kopie des zweiten Speicherblocks von dem Kohärenztransformator zu dem Speichermodul an dem Computerknoten; und

wenn die erste Kopie des ersten Speicherblocks nicht verfolgt worden ist in einem Schnüffel-Tag des Schnüffel-Tag-Arrays, Modifizieren unter Verwendung des Kohärenztransformators eines Mtags, das dem zweiten Speicherblock in dem Speichermodul bei dem Computerknoten zugeordnet ist zum Reflektieren, dass der Computerknoten eine ausschließliche Kopie des zweiten Speicherblocks hat.

8. Verfahren nach Anspruch 1, wobei der globale Zustand für die jeweiligen der Speicherblöcke verwendet wird als externer Zustand für die jeweiligen der Speicherblöcke, wobei ein globaler Ausschließlichkeits-Zustand einen externen Ungültigkeits-Zustand repräsentiert, ein globaler Geteilt-Zustand einen externen Geteilt-Zustand repräsentiert und ein globaler Ungültigkeits-Zustand einen externen Ausschließlichkeits-Zustand repräsentiert.

9. Verfahren nach Anspruch 1, außerdem die Schritte umfassend:

Empfangen einer Rückschreibanforderung für den ersten Speicherblock von der externen Einrichtung bei dem Kohärenztransformator;

Erhalten der ersten Kopie des ersten Speicherblocks unter Verwendung des Kohärenztransformators von der externen Einrichtung;

Schreiben der ersten Kopie des ersten Speicherblocks von dem Kohärenztransformator zu dem Speichermodul an dem Computerknoten; und

wenn die erste Kopie des ersten Speicherblocks in einem Schnüffel-Tag des Schnüffel-Tag-Arrays verfolgt worden ist, vor dem Schreibschritt, Aufheben der Zuordnung des Schnüffel-Tags des Schnüffel-Tag-Arrays, hierdurch das Schnüffel-Tag verfügbar machend zum Verfolgen anderer extern gecacheter Speicherblöcke und Veranlassen, dass die erste Kopie des ersten Speicherblocks nicht länger verfolgt wird durch das Schnüffel-Tag-Array.

10. Verfahren nach Anspruch 1, außerdem den Schritt umfassend des Ansprechens durch den Kohärenztransformator auf eine zweite Speicherzugriffsanforderung auf dem gemeinsamen Bus im Auftrag der externen Einrichtung, umfassend:

Überwachen der Speicherzugriffsanforderungen auf dem gemeinsamen Bus, Verwenden des Kohärenztransformators zum Bestimmen, ob eine zweite Speicherzugriffsanforderung der Speicherzugriffsanforderungen auf dem gemeinsamen Bus zu irgendeinem der in Snoop-Tags des Snoop-Tag-Arrays verfolgten Speicherblöcke gehört; und

wenn die zweite Speicherzugriffsanforderung zu einem zweiten Speicherblock gehört, wobei der zweite Speicherblock den einen der in den Schnüffel-Tags des Schnüffel-Tag-Arrays verfolgten Speicherblöcken repräsentiert, ansprechend auf die zweite Speicherzugriffsanforderung unter Verwendung der Nur-Schnüffel-Methode, einschließlich des Ansprechens auf die zweite Speicherzugriffsanforderung, unter Verwendung des Kohärenztransformators.

11. Verfahren nach Anspruch 10, wobei der Kohärenztransformator auf die zweite Speicherzugriffsanforderung in der Nur-Schnüffel-Methode nur anspricht, wenn ein Schnüffel-Tag, das den zweiten Speicherblock in dem Schnüffel-Tag-Array verfolgt, anzeigt, dass eine erste Kopie des zweiten Speicherblocks gültig ist in der externen Einrichtung.

12. Verfahren nach Anspruch 11, wobei die zweite Speicherzugriffsanforderung eine Anforderung für eine ausschließliche Kopie ist und das Schnüffel-Tag, das den zweiten Speicherblock verfolgt, anzeigt, dass die erste Kopie des zweiten Speicherblocks bei der externen Einrichtung eine ausschließliche Kopie des zweiten Speicherblocks ist, wobei der Schritt des Ansprechens auf die zweite Speicherzugriffsanforderung unter Verwendung der Nur-Schnüffel-Methode umfasst:

Erhalten einer zweiten Kopie des zweiten Speicherblocks unter Verwendung des Kohärenztransformators von der ersten Kopie des zweiten Speicherblocks bei der externen Einrichtung;

Ungültigmachen der ersten Kopie des zweiten Speicherblocks bei der externen Einrichtung; und

Weiterleiten der zweiten Kopie des zweiten Speicherblocks von dem Kohärenztransformator zu dem gemeinsamen Bus zum Befähigen eines Erzeugers der zweiten Speicherzugriffsanforderung, die zweite Kopie des zweiten Speicherblocks zu erhalten; und

Aufheben der Zuordnung des Schnüffel-Tags des Schnüffel-Tag-Arrays, hierdurch verfügbar machen des Schnüffel-Tags zum Verfolgen anderer extern gecacheter Speicherblöcke und Veranlassen des zweiten Speicherblocks, nicht länger verfolgt zu werden durch das Schnüffel-Tag-Array.

13. Verfahren nach Anspruch 11, wobei die zweite Speicherzugriffsanforderung eine Anforderung für eine ausschließliche Kopie ist und das den zweiten Speicherblock verfolgende Schnüffel-Tag anzeigt, dass die erste Kopie des zweiten Speicherblocks bei der externen Einrichtung eine geteilte Kopie des zweiten Speicherblocks ist, wobei der Schritt des Ansprechens auf die zweite Speicherzugriffsanforderung unter Verwendung der Nur-Schnüffel-Methode umfasst:

Ungültigmachen der ersten Kopie des zweiten Speicherblocks bei der externen Einrichtung;

Erhalten einer zweiten Kopie des zweiten Speicherblocks unter Verwendung des Kohärenztransformators von dem Computerknoten über den gemeinsamen Bus;

Ungültigmachen irgendeiner gültigen Kopie des zweiten Speicherblocks unter Verwendung des Kohärenztransformators in dem Computerknoten; und

Weiterleiten der zweiten Kopie des zweiten Speicherblocks von dem Kohärenztransformator zu dem gemeinsamen Bus, um einen Erzeuger der Seitenspeicherzugriffsanforderung in die Lage zu versetzen, die zweite Kopie des zweiten Speicherblocks zu erhalten; und

Aufheben der Zuordnung des Schnüffel-Tags des Schnüffel-Tag-Arrays, hierdurch das Schnüffel-Tag verfügbar machend zum Verfolgen anderer extern gecacheter Speicherblöcke und Veranlassen, dass der zweite Speicherblock nicht länger verfolgt wird von dem Schnüffel-Tag-Array.

14. Verfahren nach Anspruch 11, wobei die zweite Speicherzugriffsanforderung eine Anforderung für eine geteilte Kopie ist und das Schnüffel-Tag, das den zweiten Speicherblock verfolgt, anzeigt, dass die erste Kopie des zweiten Speicherblocks bei der externen Einrichtung eine geteilte Kopie des zweiten Speicherblocks ist, wobei der Schritt des Ansprechens auf die zweite Speicherzugriffsanforderung unter Verwendung der Nur-Schnüffel-Methode umfasst:

Erhalten einer zweiten Kopie des zweiten Speicherblocks unter Verwendung des Kohärenztransformators von dem Computerknoten über den gemeinsamen Bus; und

Weiterleiten der zweiten Kopie des zweiten Speicherblocks von dem Kohärenztransformator zu dem gemeinsamen Bus, um den Erzeuger der zweiten Speicherzugriffsanforderung in die Lage zu versetzen, die zweite Kopie des zweiten Speicherblocks zu erhalten.

15. Verfahren nach Anspruch 11, wobei die zweite Speicherzugriffsanforderung eine Anforderung für eine geteilte Kopie ist und das Schnüffel-Tag, das den zweiten Speicherblock verfolgt, anzeigt, dass die erste Kopie des zweiten Speicherblocks bei der externen Einrichtung eine ausschließliche Kopie des zweiten Speicherblocks ist, wobei der Schritt des Ansprechens auf die zweite Speicherzugriffsanforderung unter Verwendung der Nur-Schnüffel-

Methode umfasst:

Erhalten einer zweiten Kopie des zweiten Speicherblocks von der externen Einrichtung unter Verwendung des Kohärenztransformators;

Ändern des Schnüffel-Tags, das den zweiten Speicherblock verfolgt zum Anzeigen, dass die erste Kopie des Speicherblocks bei der externen Einrichtung eine geteilte Kopie des zweiten Speicherblocks ist; und

Weiterleiten der zweiten Kopie des zweiten Speicherblocks von dem Kohärenztransformator zum gemeinsamen Bus, um einen Erzeuger der zweiten Speicherzugriffsanforderung in die Lage zu versetzen, die zweiten Kopie des zweiten Speicherblocks zu erhalten.

16. Kohärenztransformator (200) zum Erleichtern des Teilens von Speicherblöcken (112) zwischen einem Computerknoten (100) und einer externen Einrichtung, wobei der Computerknoten einen gemeinsamen Bus (108) einschließt, mit dem der Kohärenztransformator, ein Speichermodul (110) und ein Verarbeitungsknoten (102, 104, 106) mit einem Prozessor und einem Cache (114) verbunden sind, wobei die Speicherblöcke lokale physikalische Adressen in dem Speichermodul an dem Computerknoten haben, jeder der Speicherblöcke ein zugeordnetes Speicheretikett bzw. Mtag zum Verfolgen eines globalen Zustandes hat, der jedem Speicherblock zugeordnet ist, einschließlich eines globalen ausschließlichen Zustandes zum Anzeigen, dass der Speicherblock ausschließlich für den Computerknoten ist, eines globalen geteilten Zustandes zum Anzeigen, dass der Speicherblock geteilt ist durch den Computerknoten mit der externen Einrichtung und eines globalen Ungültigkeits-Zustandes zum Anzeigen, dass der jeweilige Speicherblock ungültig ist in dem Computerknoten, wobei der Kohärenztransformator umfasst:

eine Schnüffel-Logik (260), die konfiguriert ist zum Koppeln mit dem gemeinsamen Bus des Computerknotens, wobei die Schnüffel-Logik, wenn sie mit dem gemeinsamen Bus gekoppelt ist, betreibbar ist zum Überwachen von Speicherzugriffsanforderungen auf dem gemeinsamen Bus;

wobei der Kohärenztransformator **gekennzeichnet ist durch** das Umfassen:

eines Schnüffel-Tag-Arrays (250), das mit der Schnüffel-Logik gekoppelt ist, wobei das Schnüffel-Tag-Array eine Vielzahl von Schnüffel-Tags (273, 274, 276, 278, 280) hat, jedes der Vielzahl von Schnüffel-Tags konfiguriert ist zum Identifizieren eines der Speicherblöcke, wenn er von der externen Einrichtung gecached ist und zum Verfolgen eines externen Zustandes einer Kopie des Speicherblocks, wobei der externe Zustand einen einschließt von einem externen Ausschließlichkeits-Zustand zum Anzeigen, dass die Kopie dieses Speicherblocks ausschließlich für die externe Domain ist, eines externen geteilten Zustandes zum Anzeigen, dass die Kopie dieses Speicherblocks geteilt ist von der externen Domain und eines externen Ungültigkeits-Zustandes zum Anzeigen, dass die Kopie dieses Speicherblocks ungültig ist in der externen Domain; und

eine Logikvorrichtung zum Ermitteln (504, 506, 508), ob eine erste Speicherzugriffsanforderung von der externen Einrichtung zum Cachen eines ersten Speicherblocks beantworten werden sollte unter Verwendung einer Nur-Schnüffel-Methode, in der ein Tag in dem Schnüffel-Tag-Array betreibbar ist zum Verfolgen des externen Zustandes einer Kopie des ersten Speicherblocks für eine gesamte Dauer, die der erste Speicherblock von der externen Einrichtung gecached ist oder Verwendung einer Nur-Mtag-Methode, bei der ein Etikett bzw. Tag für den ersten Speicherblock temporär gespeichert wird, bis ein globaler Zustand, der dem ersten Speicherblock zugeordnet ist, zurückgeschrieben werden kann in das Speichermodul.

17. Kohärenztransformator nach Anspruch 16, außerdem eine Logik (606, 608, 610) umfassend zum Feststellen, ob ein zweiter Speicherzugriff für einen zweiten Speicherblock auf dem gemeinsamen Bus beantwortet werden sollte unter Verwendung der Nur-Schnüffel-Methode oder der Nur-Mtag-Methode, wobei der zweiten Speicherzugriff beantwortet wird durch den Kohärenztransformator unter Verwendung der Nur-Schnüffel-Methode, wenn der zweite Speicherblock von dem Schnüffel-Tag-Array verfolgt wird, wobei durch das Speichermodul auf den zweiten Speicherblock angesprochen wird unter Verwendung der Nur-Mtag-Methode, wenn der zweite Speicherzugriff nicht von dem Schnüffel-Tag-Array verfolgt wird.

18. Computersystem mit einem Computerknoten (100), dem Kohärenztransformator (200) nach Anspruch 16 oder Anspruch 17 und einer externen Einrichtung, wobei der Computerknoten einen gemeinsamen Bus (108) einschließt, an den der Kohärenztransformator (200), ein Speichermodul (110) und ein Verarbeitungsknoten (102,

104, 106) mit einem Prozessor (116) und einem Cache (114) verbunden sind.

Revendications

1. Procédé pour permettre à un dispositif externe (202, 204, 206) dans un domaine externe qui est externe à un noeud informatique (100) d'un système informatique, de partager des blocs de mémoire (112) ayant des adresses physiques locales dans un module de mémoire (110) audit noeud informatique indépendamment du fait que ledit dispositif externe et le bus commun (108) audit noeud informatique utilisent tous deux un protocole commun et indépendamment du fait que ledit dispositif externe et ledit bus commun opèrent tous deux à la même vitesse, ledit noeud informatique comportant un transformateur de cohérence (200), ledit module de mémoire et un noeud de traitement connecté audit bus commun, ledit noeud de traitement (102, 104, 106) ayant un processeur (116) et un cache (114), chacun desdits blocs de mémoire ayant une balise de mémoire associée (Mtag) pour pister un état global associé à chaque bloc mémoire, incluant un état exclusif global pour indiquer que chaque bloc mémoire est exclusif audit noeud informatique, à l'état partagé global pour indiquer que chaque bloc mémoire est partagé par ledit noeud informatique et ledit dispositif externe, et un état invalide global pour indiquer que chaque bloc mémoire est invalide dans ledit noeud informatique, ledit procédé comprenant les étapes consistant à :

   mettre sous surveillance de trafic ledit bus commun afin de contrôler les demandes d'accès à la mémoire sur ledit bus commun ;
   recevoir, au transformateur de cohérence, une première demande d'accès à la mémoire pour mettre en antémémoire un premier bloc de mémoire à partir dudit dispositif externe ;
   obtenir une première copie dudit bloc de mémoire, utiliser ledit transformateur de cohérence, à partir dudit bus commun, **caractérisé en ce que** ledit transformateur de cohérence comporte un tableau de balises de surveillance de trafic (250) ayant une pluralité de balises de surveillance de trafic, chacune de ladite pluralité de balises de surveillance de trafic étant configurée pour identifier un desdits blocs de mémoire si ce dernier est mis en antémémoire par ledit dispositif externe et pour pister un état externe d'une copie de ce bloc de mémoire, ledit état externe incluant l'un d'un état exclusif externe pour indiquer que ladite copie de ce bloc de mémoire est exclusif audit domaine externe, d'un état partagé externe pour indiquer que ladite copie de ce bloc de mémoire est partagée par ledit domaine externe, et d'un état invalide externe pour indiquer que ladite copie de ce bloc de mémoire est invalide dans ledit domaine externe ; et
   si au moins une balise de ladite pluralité de balises de surveillance de trafic est disponible pour pister ledit état externe de ladite première copie dudit premier bloc de mémoire, répondre à ladite première demande d'accès à la mémoire en utilisant une approche par surveillance de trafic seulement dans laquelle ladite balise est utilisée pour pister ledit état externe de ladite première copie dudit premier bloc de mémoire pour une durée entière, que ledit premier bloc de mémoire est mis en antémémoire par ledit dispositif externe ;
   autrement, si au moins une des balises de ladite pluralité de balises de surveillance de trafic n'est pas disponible pour pister ledit état externe de ladite première copie dudit premier bloc de mémoire, répondre à ladite première demande d'accès à la mémoire en utilisant une approche par balise de mémoire (Mtag) seulement dans laquelle, en utilisant ledit transformateur de cohérence, une balise pour ledit premier bloc de mémoire est temporairement stockée jusqu'à ce qu'un stade global associé audit premier bloc de mémoire puisse être réinscrit dans ledit module de mémoire ;
   ladite première copie dudit premier bloc de mémoire étant envoyée dudit transformateur de cohérence audit dispositif externe.

2. Procédé selon la revendication 1, dans lequel ladite première demande d'accès à la mémoire à partir dudit dispositif externe représente une demande pour une copie exclusive dudit premier bloc de mémoire et ladite étape de réponse à ladite première demande d'accès à la mémoire utilisant ladite approche par balise de mémoire seulement comporte de plus une étape consistant à changer ladite première balise de mémoire dans ledit module de mémoire en un état invalide global.

3. Procédé selon la revendication 2, dans lequel ladite étape de réponse à ladite première demande d'accès à la mémoire utilisant ladite approche par balise de mémoire seulement comporte de plus une étape d'invalidation de toutes les copies valides dudit premier bloc de mémoire audit noeud informatique.

4. Procédé selon la revendication 1, dans lequel ladite première demande d'accès à la mémoire à partir dudit dispositif externe représente soit une demande pour une copie exclusive dudit premier bloc de mémoire, soit une demande pour une copie partagée dudit premier bloc de mémoire, l'étape de réponse à ladite première demande d'accès

à la mémoire utilisant ladite approche par balise de mémoire seulement comporte de plus les étapes consistant à :

avant ladite étape de modification, examiner ladite première balise de mémoire (Mtag) associée audit premier bloc de mémoire; et
ne procéder à ladite étape de modification et à ladite étape d'envoi que si la première balise de mémoire (Mtag) ne représente pas un état invalide global.

5. Procédé selon la revendication 1, dans lequel ladite premlère demande d'accès à la mémoire à partir dudit dispositif externe représente une demande pour une copie partagée dudit premier bloc de mémoire et ladite étape de réponse à ladite première demande d'accès à la mémoire utilisant ladite approche par balise de mémoire seulement comporte de plus une étape de modification de ladite première balise de mémoire dans ledit module de mémoire en un état partagé global.

6. Procédé selon la revendication 5, dans lequel ladite étape de réponse à ladite première demande d'accès à la mémoire utilisant ladite approche par balise de mémoire seulement comporte de plus les étapes consistant à :

avant ladite étape de modification, examiner ladite première balise de mémoire (Mtag) associée audit premier bloc de mémoire; et
ne procéder à ladite étape de modification et à ladite étape d'envoi que si ladite première balise de mémoire (Mtag) ne représente pas un état invalide global ; et
si ladite première balise de mémoire (Mtag) représente un état invalide global, signaler une condition d'erreur.

7. Procédé selon la revendication 1, comprenant de plus les étapes consistant à :

recevoir une demande de réinscription pour un second bloc de mémoire à partir dudit dispositif externe audit transformateur de cohérence ;
obtenir ladite première copie dudit second bloc de mémoire, en utilisant ledit transformateur de cohérence à partir dudit dispositif externe ;
écrire ladite première copie dudit second bloc de mémoire à partir dudit transformateur de cohérence vers ledit module de mémoire audit noeud informatique ; et
si ladite première copie dudit premier bloc de mémoire n'est pas pistée dans une balise de surveillance de trafic dudit tableau de balises de surveillance de trafic, modifier, en utilisant ledit transformateur de cohérence, une balise de mémoire (Mtag) associée audit second bloc de mémoire dans ledit module de mémoire audit noeud informatique pour refléter le fait que ledit noeud informatique a une copie exclusive dudit second bloc de mémoire.

8. Procédé selon la revendication 1, dans lequel ledit état global pour ledit chacun desdits blocs de mémoire est employé en tant que ledit état externe pour ledit chacun desdits blocs de mémoire, de manière qu'un état exclusif global représente un état invalide externe, un état partagé global représente un état partagé externe, et un état invalide global représente un état exclusif externe.

9. Procédé selon la revendication 1, comprenant de plus les étapes consistant à :

recevoir une demande de réinscription pour ledit premier bloc de mémoire à partir dudit dispositif externe audit transformateur de cohérence ;
obtenir ladite première copie dudit premier bloc de mémoire, en utilisant le transformateur de cohérence, à partir dudit dispositif externe ;
écrire ladite première copie dudit premier bloc de mémoire à partir dudit transformateur de cohérence vers ledit module de mémoire audit noeud informatique ; et
si ladite première copie dudit premier bloc de mémoire a été pistée dans une balise de surveillance de trafic dudit tableau de balises de surveillance de trafic avant ladite étape d'écriture, désallouer ladite balise de surveillance de trafic dudit tableau de balises de surveillance de trafic, de manière à rendre ladite balise de surveillance de trafic disponible pour le pistage d'autres blocs de mémoire mis en antémémoire extérieurement et faire en sorte que ladite première copie dudit premier bloc de mémoire ne soit plus pistée par ledit tableau de balises de surveillance de trafic.

10. Procédé selon la revendication 1, comprenant de plus l'étape de réponse, par ledit transformateur de cohérence, à une seconde demande d'accès à la mémoire sur ledit bus commun de la part dudit dispositif externe, comprenant

les étapes consistant à :

contrôler les demandes d'accès à la mémoire sur ledit bus commun en utilisant ledit transformateur de cohérence, pour déterminer si une seconde demande d'accès à la mémoire desdites demandes d'accès à la mémoire sur ledit bus commun concerne l'un quelconque des blocs de mémoire pisté dans les balises de surveillance de trafic dudit tableau de balises de surveillance de trafic ; et
si ladite seconde demande d'accès à la mémoire concerne un second bloc de mémoire, ledit second bloc de mémoire représentant ledit un des blocs de mémoire pisté dans lesdites balises de surveillance de trafic dudit tableau de balises de surveillance de trafic, répondre à ladite seconde demande d'accès à la mémoire en utilisant ladite approche par balise de surveillance de trafic seulement, y compris répondre à ladite seconde demande d'accès à la mémoire en utilisant ledit transformateur de cohérence.

11. Procédé selon la revendication 10, dans lequel ledit transformateur de cohérence ne répond à ladite seconde demande d'accès à la mémoire dans ladite approche par balise de surveillance de trafic seulement, que si une balise de surveillance de trafic pistant ledit second bloc de mémoire dans ledit tableau de balises de surveillance de trafic indique qu'une première copie dudit second bloc de mémoire est valide audit dispositif externe.

12. Procédé selon la revendication 11, dans lequel ladite seconde demande d'accès à la mémoire est une demande pour une copie exclusive et ladite balise de surveillance de trafic pistant ledit second bloc de mémoire indique que ladite première copie dudit second bloc de mémoire audit dispositif externe est une copie exclusive dudit second bloc de mémoire, ladite étape de réponse à ladite seconde demande d'accès à la mémoire comprend les étapes consistant à :

obtenir, en utilisant le transformateur de cohérence, une seconde copie dudit second bloc de mémoire à partir de ladite première copie dudit second bloc de mémoire audit dispositif externe ;
invalider ladite première copie dudit second bloc de mémoire de mémoire audit dispositif externe ; et
réacheminer ladite seconde copie dudit second bloc de mémoire depuis ledit transformateur de cohérence vers ledit bus commun pour permettre à un initiateur de ladite seconde demande d'accès à la mémoire d'obtenir ladite seconde copie dudit second bloc de mémoire ; et
désallouer ladite balise de surveillance de trafic dudit tableau de balises de surveillance de trafic, de manière à rendre ladite balise de surveillance de trafic disponible pour le pistage d'autres blocs de mémoire mis en antémémoire extérieurement et pour faire en sorte que ledit second bloc de mémoire ne soit plus pisté par ledit tableau de balises de surveillance de trafic.

13. Procédé selon la revendication 11, dans lequel ladite seconde demande d'accès à la mémoire est une demande pour une copie exclusive et ladite balise de surveillance de trafic pistant ledit second bloc de mémoire indique que ladite première copie dudit second bloc de mémoire audit dispositif externe est une copie partagée dudit second bloc de mémoire, ladite étape de réponse à ladite seconde demande d'accès à la mémoire utilisant ladite approche par balise de surveillance de trafic seulement, comprenant les étapes consistant à :

invalider ladite première copie dudit second bloc de mémoire audit dispositif externe ;
obtenir, en utilisant le transformateur de cohérence, une seconde copie dudit second bloc de mémoire à partir dudit noeud informatique via ledit bus commun ;
invalider, en utilisant ledit transformateur de cohérence, toute copie valide dudit second bloc de mémoire dans ledit noeud informatique ; et
réacheminer ladite seconde copie dudit second bloc de mémoire depuis ledit transformateur de cohérence vers ledit bus commun pour autoriser un initiateur de ladite seconde demande d'accès à la mémoire à obtenir ladite seconde copie dudit second bloc de mémoire ; et
désallouer ladite balise de surveillance de trafic dudit tableau de balises de surveillance de trafic, de manière à rendre ladite balise de surveillance de trafic disponible pour le pistage d'autres blocs de mémoire mis en antémémoire extérieurement, et pour faire en sorte que ledit second bloc de mémoire ne soit plus pisté par ledit tableau de balises de surveillance de trafic.

14. Procédé selon la revendication 11, dans lequel ladite seconde demande d'accès à la mémoire est une requête pour une copie partagée et ladite balise de surveillance de trafic pistant ledit second bloc de mémoire indique que ladite première copie dudit second bloc de mémoire audit dispositif externe est une copie partagée dudit second bloc de mémoire, ladite étape de réponse à ladite seconde demande d'accès à la mémoire utilisant ladite approche par balise de surveillance de trafic seulement, comprenant les étapes consistant à :

obtenir, en utilisant ledit transformateur de cohérence, une seconde copie dudit second bloc de mémoire à partir dudit noeud informatique via ledit bus commun ; et
réacheminer ladite seconde copie dudit second bloc de mémoire depuis ledit transformateur de cohérence vers ledit bus commun pour autoriser un initiateur de ladite seconde demande d'accès à la mémoire à obtenir ladite seconde copie dudit second bloc de mémoire.

15. Procédé selon la revendication 11, dans lequel ladite seconde demande d'accès à la mémoire est une requête pour une copie partagée et ladite balise de surveillance de trafic pistant ledit second bloc de mémoire indique que ladite première copie dudit second bloc de mémoire audit dispositif externe est une copie exclusive dudit second bloc de mémoire, ladite étape de réponse à ladite seconde demande d'accès à la mémoire utilisant ladite approche par balise de surveillance de trafic seulement, comprenant les étapes consistant à :

obtenir, en utilisant ledit transformateur de cohérence, une seconde copie dudit second bloc de mémoire à partir dudit dispositif externe ;
changer ladite balise de surveillance de trafic pistant ledit second bloc de mémoire pour indiquer que ladite première copie dudit second bloc de mémoire audit dispositif externe est une copie partagée dudit second bloc de mémoire ; et
envoyer ladite seconde copie dudit second bloc de mémoire depuis ledit transformateur de cohérence vers ledit bus commun afin d'autoriser un initiateur de ladite seconde demande d'accès à la mémoire à obtenir ladite seconde copie dudit second bloc de mémoire.

16. Transformateur de cohérence (200) pour faciliter le partage de blocs de mémoire (112) entre un noeud informatique (100) et un dispositif externe, ledit noeud informatique comportant un bus commun (108) auquel ledit transformateur de cohérence, un module de mémoire (110) et un noeud de traitement (102, 104, 106) avec un processeur et un cache (114) sont connectés, lesdits blocs de mémoire ayant des adresses physiques locales dans le module de mémoire audit noeud informatique, chacun desdits blocs de mémoire ayant une balise de mémoire (Mtag) associée pour pister un état global associé à chaque bloc mémoire, incluant un état exclusif global pour indiquer que ce bloc mémoire est exclusif audit noeud informatique, un état partagé global pour indiquer que le bloc mémoire est partagé par ledit noeud informatique et ledit dispositif externe, et un état invalide global pour indiquer que chaque bloc mémoire est invalide dans ledit noeud informatique, ledit transformateur de cohérence comprenant :

une logique de surveillance de trafic (260) configurée pour être couplée audit bus commun dudit noeud informatique, ladite logique de surveillance de trafic, lorsqu'elle est couplée audit bus commun, pouvant être mise en oeuvre pour contrôler des demandes d'accès à la mémoire sur ledit bus commun ;
ledit transformateur de cohérence étant **caractérisé en ce qu**'il comprend :

un tableau de balises de surveillance de trafic (250) couplé à ladite logique de surveillance de trafic, ledit tableau de balises de surveillance de trafic ayant une pluralité de balises de surveillance de trafic (273, 274, 276, 278, 280), chacune de ladite pluralité de balises de surveillance de trafic étant configurée pour identifier un desdits blocs de mémoire s'il est mis en antémémoire par ledit dispositif externe, et pour pister un état externe d'une copie de ce bloc de mémoire, ledit état externe incluant un d'un état exclusif externe pour indiquer que ladite copie de ce bloc mémoire est exclusive audit domaine externe, d'un état partagé externe pour indiquer que ladite copie de ce bloc de mémoire est partagée par ledit domaine externe, et d'un état invalide externe pour indiquer que ladite copie de ce bloc de mémoire est invalide dans ledit domaine externe ; et
des moyens logiques pour déterminer (504, 506, 508) si une première demande d'accès à la mémoire à partir dudit dispositif externe pour mettre en antémémoire un premier bloc de mémoire doit être l'objet d'une réponse en utilisant une approche par balise de surveillance de trafic seulement dans laquelle une balise dudit tableau de balises de surveillance de trafic peut être mise en oeuvre pour pister ledit état externe d'une copie dudit premier bloc de mémoire pour une durée entière, selon laquelle ledit premier bloc de mémoire est mis en antémémoire par ledit dispositif externe, ou en utilisant une approche par balise de mémoire (Mtag) seulement dans laquelle une balise pour ledit premier bloc de mémoire est temporairement stockée jusqu'à ce qu'un état global associé audit premier bloc de mémoire peut être réinscrit dans ledit module de mémoire.

17. Transformateur de cohérence selon la revendication 16, comprenant de plus une logique (606, 608, 610) pour déterminer si un second accès à la mémoire pour un second bloc de mémoire sur ledit bus commun doit faire l'objet d'une réponse en utilisant ladite approche par balise de surveillance de trafic seulement ou ladite approche

par balise de mémoire (Mtag) seulement, ledit second accès à la mémoire faisant l'objet d'une réponse par ledit transformateur de cohérence en utilisant ladite approche par balise de surveillance de trafic seulement lorsque ledit second bloc de mémoire est pisté par ledit tableau de balises de surveillance de trafic, ledit second bloc de mémoire faisant l'objet d'une réponse par ledit module de mémoire en utilisant ladite approche par balise de mémoire (Mtag) seulement lorsque ledit second accès à la mémoire n'est pas pisté par ledit tableau de balises de surveillance de trafic.

18. Système informatique comportant un noeud informatique (100), ledit transformateur de cohérence (200) selon la revendication 16 ou la revendication 17 est un dispositif externe, ledit noeud informatique comportant un bus commun (108) auquel ledit transformateur de cohérence (200), un module de mémoire (110) et un noeud de traitement (102, 104, 106) avec un processeur (116) et un cache (114) sont connectés.

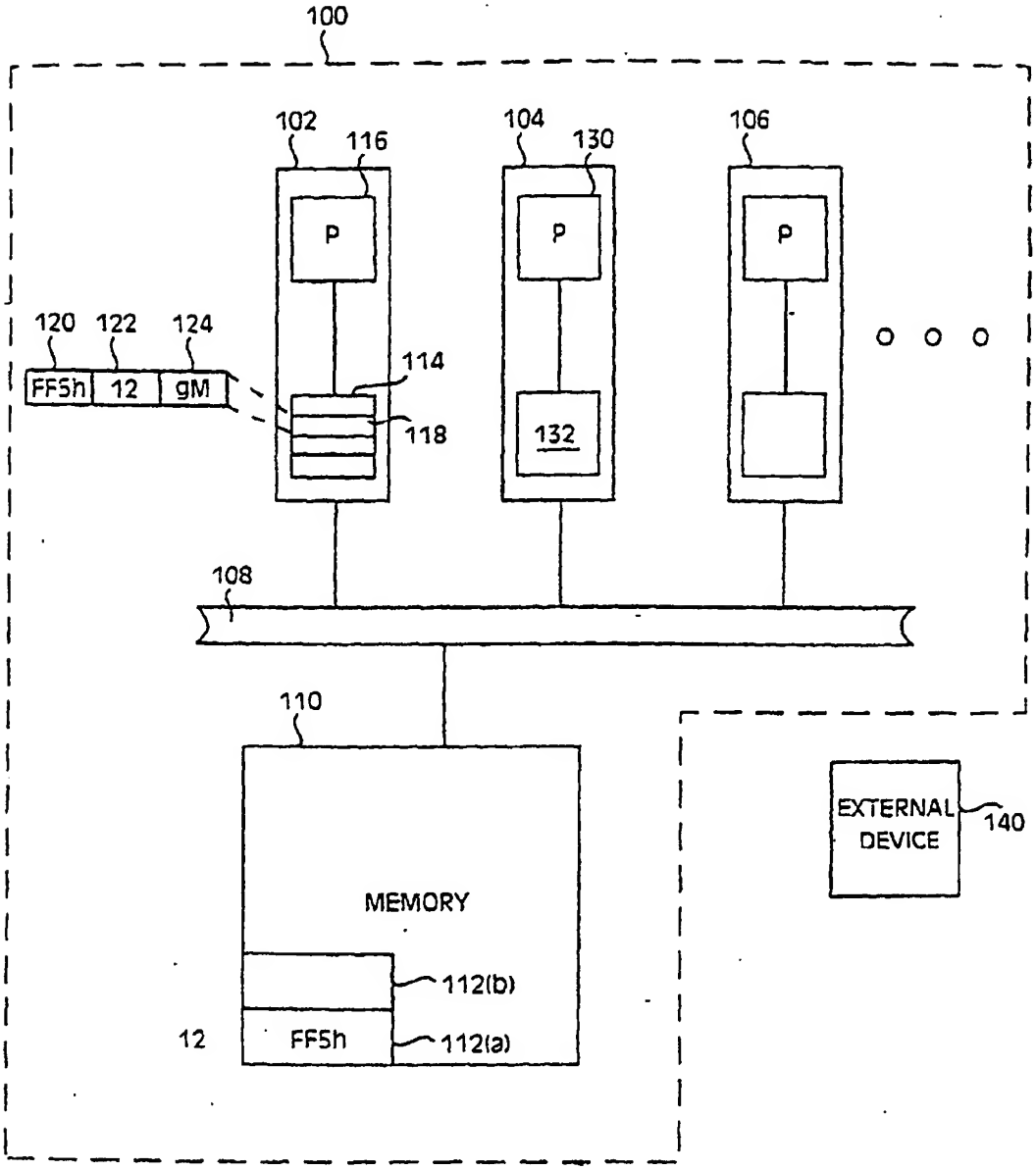
100
102
116
104
130
106
P
P
P
120
122
124
FF5h
12
gM
114
118
132
108
110
MEMORY
112(b)
12
FF5h
112(a)
EXTERNAL DEVICE
140

FIG. 1

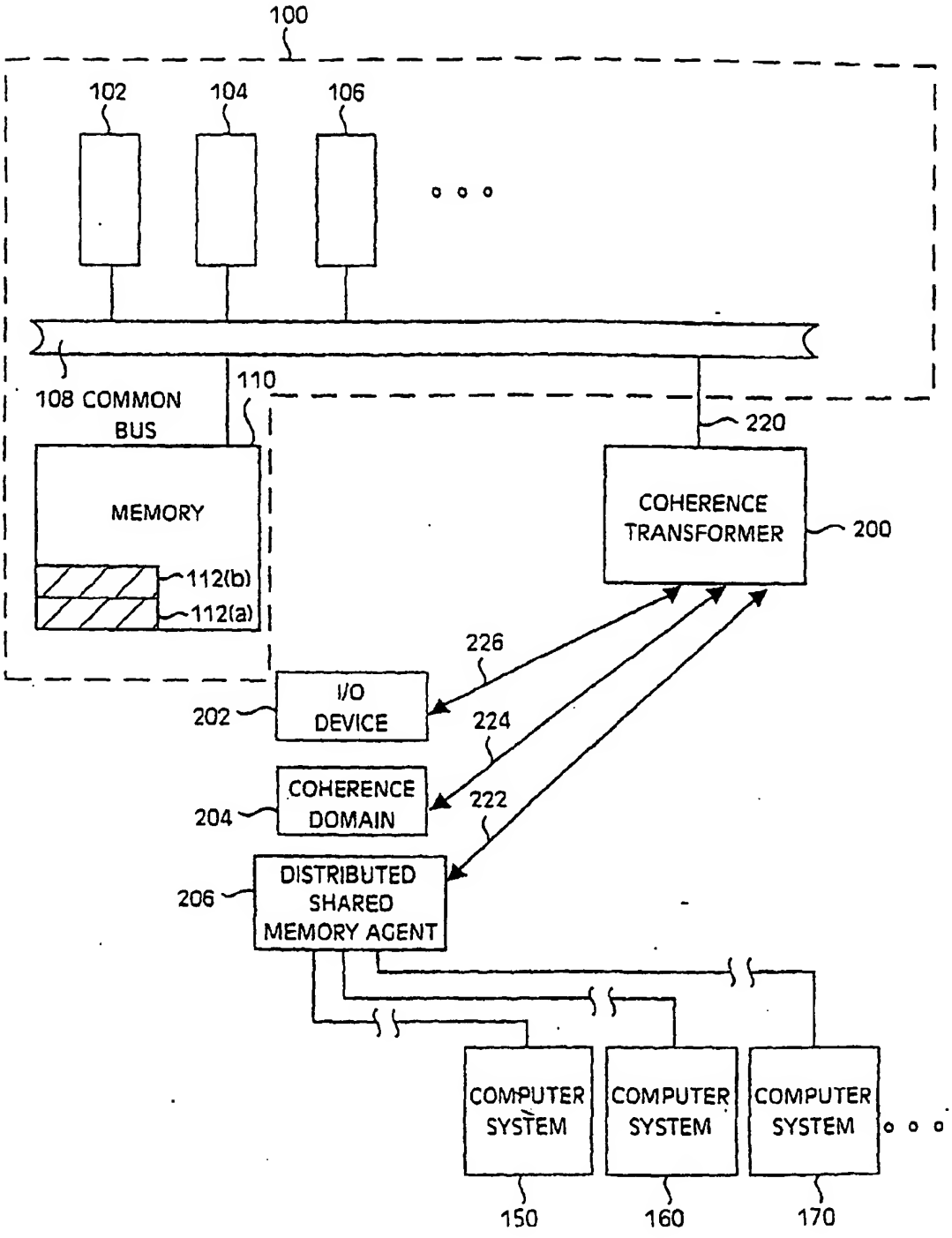
100
102
104
106
108 COMMON BUS
110
MEMORY
112(b)
112(a)
220
COHERENCE TRANSFORMER
200
226
202
I/O DEVICE
224
204
COHERENCE DOMAIN
222
206
DISTRIBUTED SHARED MEMORY AGENT
COMPUTER SYSTEM
COMPUTER SYSTEM
COMPUTER SYSTEM
150
160
170

FIG. 2

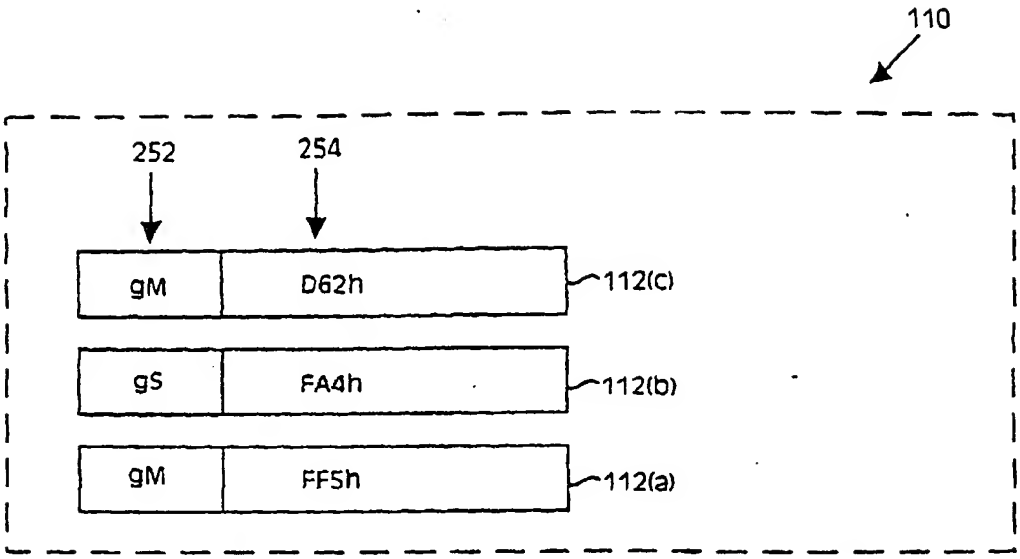

FIG. 3

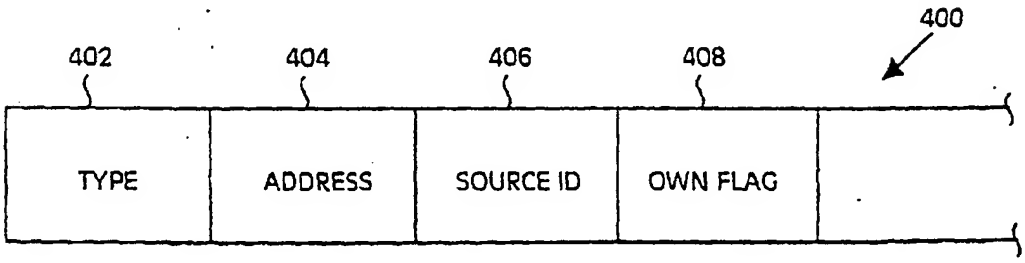

FIG. 5

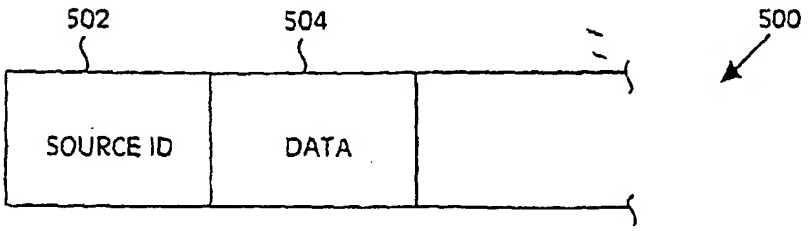

FIG. 6

STATE	REPRESENTING	MEANING
gM	Global Modified	Internal domain has a valid, exclusive (and potentially modified) copy; there are no valid external copies
gS	Global Shared	Internal domain has valid, shared copy or copies; there may be shared copy or copies externally
gI	Global Invalid	Internal domain does not have a valid (exclusive or shared) copy; there is a valid exclusive (and potentially modified) external copy

Fig. 4

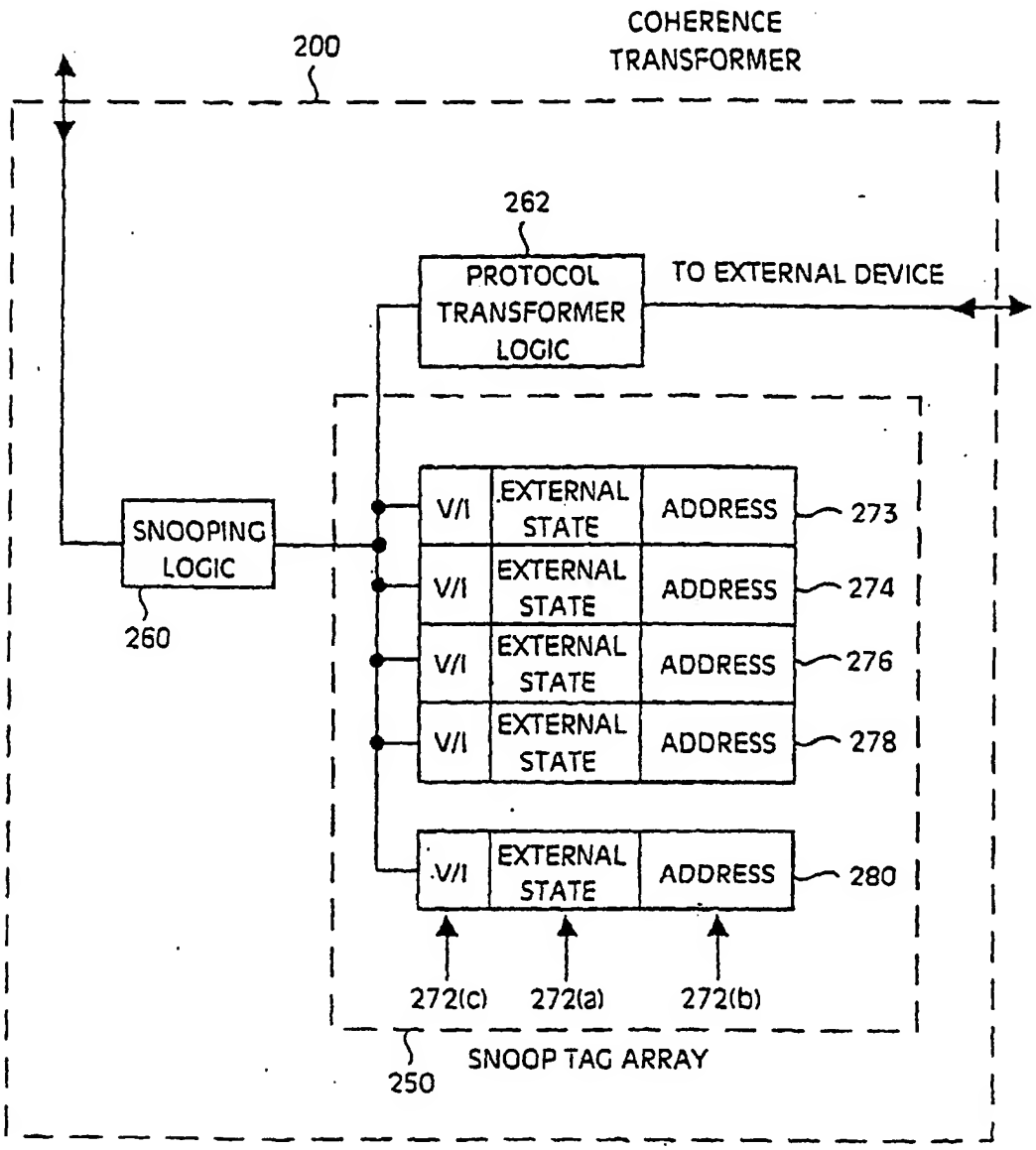
COHERENCE TRANSFORMER
200
262
PROTOCOL TRANSFORMER LOGIC
TO EXTERNAL DEVICE
SNOOPING LOGIC
260
V/I
EXTERNAL STATE
ADDRESS
273
274
276
278
280
272(c)
272(a)
272(b)
SNOOP TAG ARRAY
250

FIG. 7A

State	Representing	Meaning
eI	Invalid	CT does not have copy
eS	Shared	CT owns a shared copy
eM	Exclusive	CT owns an exclusive (potentially modified) copy

Fig. 7B

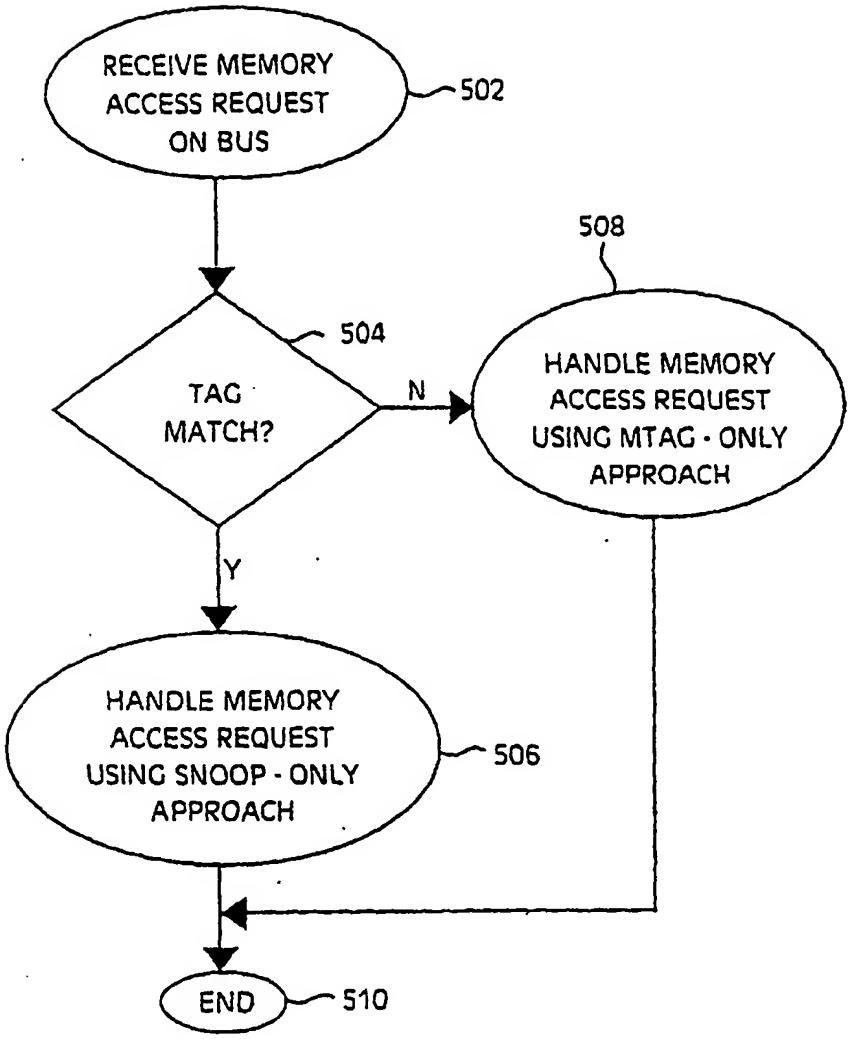
RECEIVE MEMORY ACCESS REQUEST ON BUS
502
TAG MATCH?
504
N
Y
HANDLE MEMORY ACCESS REQUEST USING MTAG - ONLY APPROACH
508
HANDLE MEMORY ACCESS REQUEST USING SNOOP - ONLY APPROACH
506
END
510

FIG. 8

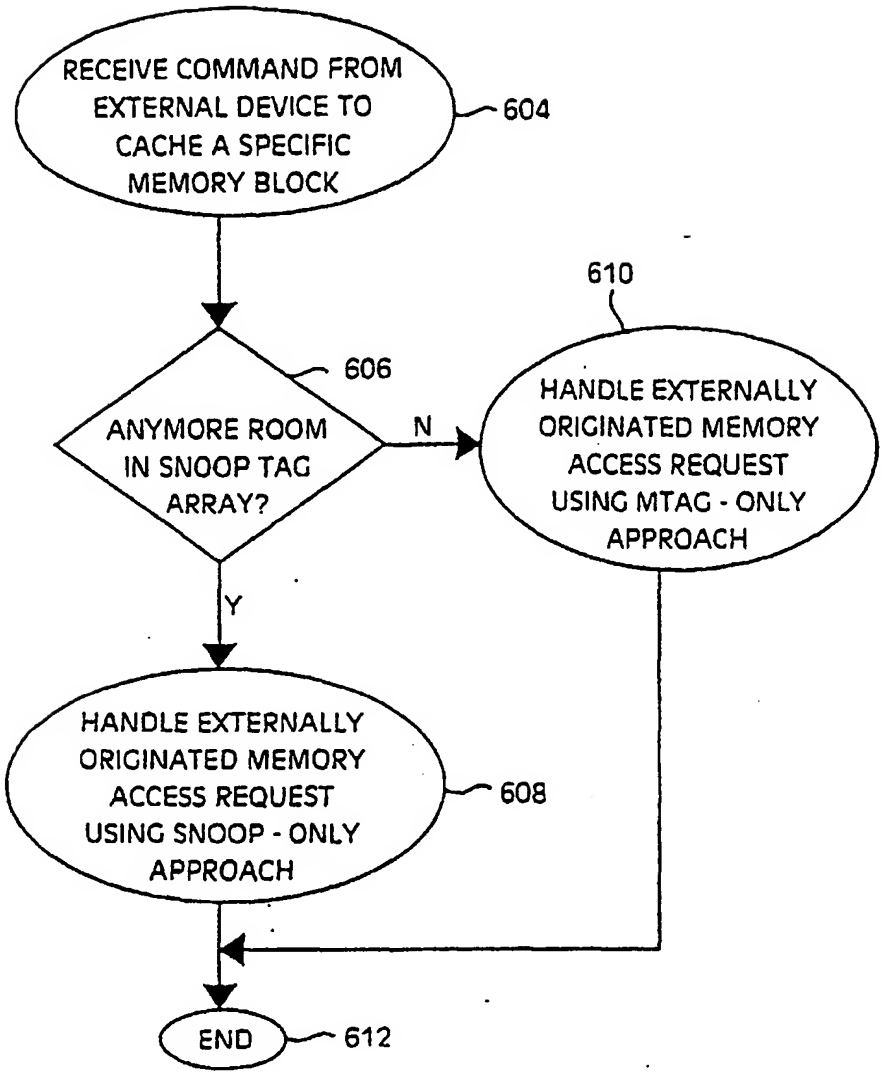
RECEIVE COMMAND FROM EXTERNAL DEVICE TO CACHE A SPECIFIC MEMORY BLOCK
604
ANYMORE ROOM IN SNOOP TAG ARRAY?
606
N
Y
HANDLE EXTERNALLY ORIGINATED MEMORY ACCESS REQUEST USING MTAG - ONLY APPROACH
610
HANDLE EXTERNALLY ORIGINATED MEMORY ACCESS REQUEST USING SNOOP - ONLY APPROACH
608
END
612

FIG. 9

Bus to CT	Current State	CT to X	X to CT	CT to Bus	Bus to Ct	New State	Comment
RTO	eI						Ignore
	eS	XINV	XINV_ack	RTO RTO_data	RTO_data	eI	
	eM	XRTO	XRTO_data	RTO_data		eI	
RTS	eI						Ignore
	eS			RTS RTS_data	RTS_data		
	eM	XRTS	XRTS_data	RTS_data		eS	
WB	eI						Ignore
	eS or eM						Error

FIG. 10

EP 0 817 069 B1

X to CT	Current State	Allocate Tag?	CT to Bus	Bus to CT	CT to X	X to CT	New State	Comments
XRTO	eI	Yes,	RTO	RTO_data	XRTO_data		eM	
	eS	No	RTO	RTO_data	XRTO_data		eM	
	eM							Error
XRTS	eI	Yes,	RTS	RTS_data	XRTS_data		eS	
	eS	No	RTS	RTS_data	XRTS_data		eS	
	eM							Error
XWB	eI or eS							Error
	eM	No	WB WB_data			XWB_data	eI	

Fig. 11

REQUEST	ACTION	POSSIBLE RESPONSES
RTO	Get exclusive copy of memory block and invalidate all other copies	RTO_data, RTO_nack
RRTO	Get exclusive copy of memory block from external side and invalidate all other copies	RTOR_data, RTOR_nack
RTS	Get shared, read-only copy of memory block	RTS_data, RTS_nack
RRTS	Get shared, read-only copy of memory block from external side	RTSR_data, RTSR_nack
WB	Request to write back currently cached exclusive copy of memory block	WB_ack, WB_nack

Fig. 12

RESPONSES	ACTION	DATA?
RTO_data	Reply with exclusive copy of memory block	Y
RTO_nack	Not acknowledged, retry RTO progenitor	N
RTOR	write a gM state to memory for the requested memory block	Y
RTOR_data	Reply with exclusive copy of memory block	Y
RTOR_nack	Not acknowledged, retry RRTO progenitor	N
RTS_data	Reply with shared copy of memory block	Y
RTS_nack	Not acknowledged, retry RTS progenitor	N
RTSR	write a gS state to memory for the requested memory block	Y
RTSR_data	Reply with shared copy of memory block	Y
RTSR_nack	Not acknowledged, retry RRTS progenitor	N
WB_nack	Not acknowledged, retry WB progenitor	Y
WB_data	Reply with data to be written back	Y

Fig. 13

Bus to CT	MTag	CT to X	X to CT	CT to Bus	Bus to CT	New MTag	Comments
RRTO	gM						Error: Ask RRTO progenitor to retry.
	gS	XINV	XINV_ack	RTO	RTO_data	gM	
				RTOR	RTOR_data		
	gI	XRTO	XRTO_data	RTOR	RTOR_data	gM	
RRTS	gM, gS						Error: Ask RRTO progenitor to retry.
	gI	XRTS	XRTS_data	RTSR	RTSR_data	gS	

Fig. 14

EP 0 817 069 B1

X to CT	MTag	CT to Bus	Bus to CT	CT to X	X to CT	New MTag	Comments
XRTO	gM gS	RTO WB WB_data	RTO_data		XRTO_data	gI	
	gI						Error
XRTS	gM, gS	RTSM	RTSM_data		XRTS_data	gS	
	gI						Error
XWB	gM, gS						Error
	gI	WSgM WSgM_data			XWB_data	gM	

Fig. 15

EP 0 817 069 B1